LilyPond

El tipografiador de música

Utilización

El equipo de desarrolladores de LilyPond

Este archivo explica cómo ejecutar los programas que se distribuyen con LilyPond versión 2.24.4. Además sugiere ciertas "buenas prácticas" para una utilización eficiente.

Para mayor información sobre la forma en que este manual se relaciona con el resto de la documentación, o para leer este manual en otros formatos, consulte Sección "Manuales" en *Información general*.

Si le falta algún manual, encontrará toda la documentación en `https://lilypond.org/`.

Para la versión de LilyPond 2.24.4

Índice General

1 Ejecutar LilyPond

Este capítulo detalla los aspectos técnicos de la ejecución de LilyPond.

1.1 Utilización normal

Casi todos los usuarios ejecutan LilyPond por medio de un interfaz gráfico; consulte Sección "Primeros pasos" en *Manual de Aprendizaje* si no lo ha leído aún. Si utiliza algún otro editor para la edición de los archivos de LilyPond, consulte la documentación de dicho programa.

1.2 Utilización desde la línea de órdenes

Esta sección contiene información adicional sobre el uso de LilyPond en la línea de órdenes. Esta forma puede ser preferible para pasarle al programa algunas opciones adicionales. Además, existen algunos programas complementarios 'de apoyo' (como `midi2ly`) que sólo están disponibles en la línea de órdenes.

Al hablar de la 'línea de órdenes', nos referimos a la consola del sistema operativo. Los usuarios de Windows posiblemente estén más familiarizados con los términos 'ventana de MS-DOS' o 'línea de comandos'; Los usuarios de MacOS X puede que estén más familiarizados con los términos 'terminal' o 'consola'. Éstos podrían requerir algunas configuraciones adicionales y deberían consultar también el apartado Sección "MacOS X" en *Información general*.

La descripción del uso de esta parte de los sistemas operativos se sale del ámbito de este manual; le rogamos que consulte otros documentos sobre este tema si no le resulta familiar la línea de órdenes.

Invocar `lilypond`

El ejecutable `lilypond` se puede llamar desde la línea de órdenes de la siguiente manera:

```
lilypond [opción]... archivo...
```

Cuando se invoca con un nombre de archivo sin extensión, se prueba en primer lugar con la extensión `.ly`. Para leer la entrada desde stdin, utilice un guión (`-`) en sustitución de *archivo*.

Cuando se procesa `archivo.ly`, la salida resultante por defecto es el archivo `archivo.pdf`. Se pueden especificar varios archivos; cada uno de ellos se procesará de forma independiente[1].

Si `archivo.ly` contiene más de un bloque `\score`, el resto de las partituras se obtiene como salida en archivos numerados, empezando por `archivo-1.pdf`. además, el valor de `output-suffix` (sufijo de salida) se inserta entre el nombre base y el número. Por ejemplo, si `archivo.ly` contiene

```
#(define output-suffix "violin")
\score { ... }
#(define output-suffix "cello")
\score { ... }
```

La salida consiste en los archivos `filename-violin.pdf` y `filename-cello-1.pdf`.

Uso de LilyPond con las posibilidades básicas del shell

Dado que LilyPond es una aplicación de consola, las posibilidades del 'shell' usado para la llamada a LilyPond también pueden aprovecharse.

Por ejemplo,

```
lilypond *.ly
```

[1] El estado de GUILE no se restablece después de procesar un archivo `.ly`, por lo que debe tener cuidado de no modificar ningún valor predeterminado desde dentro de Scheme.

procesa todos los archivos de LilyPond dentro del directorio actual.

También puede ser útil redireccionar la salida de consola (p.ej. hacia un archivo):

```
lilypond file.ly 1> salida_estandar.txt

lilypond file.ly 2> error_estandar.txt

lilypond file.ly &> todo.txt
```

Estas instrucciones redireccionan la salida 'normal', solo los 'errores' o 'todo', respectivamente, hacia un archivo de texto. Consulte la documentación de su shell concreto, Command (Windows), las aplicaciones Terminal o Console (MacOS X) para comprobar si el redireccionamiento de la salida está contemplado o si la sintaxis es distinta.

El ejemplo siguiente busca y procesa todos los archivos de entradas que estén en el directorio actual y en todos los que están por debajo de él, recursivamente. Los archivos de salida se pondrán en el mismo directorio desde el que se ejecutó la instrucción, en lugar de aquellos en los que estaban los archivos de entrada originales.

```
find . -name '*.ly' -exec lilypond '{}' \;
```

También debe funcionar para los usarios de MacOS X.

Un usuario de Windows haría lo siguiente:

```
forfiles /s /M *.ly /c "cmd /c lilypond @file"
```

introduciendo estas instrucciones desde un `indicador de órdenes` que normalmente está en `Inicio > Accessorios > Símbolo del sistema`, o escribiendo en la ventana de búsqueda 'indicador de órdenes'.

De forma alternativa, una ruta explícita al nivel superior de su carpeta que contenga todas las subcarpetas con archivos de entrada en su interior se puede especificar mediante la opción `/p`;

```
forfiles /s /p C:\Documentos\MisPartituras /M *.ly /c "cmd /c lilypond @file"
```

Si el nombre de la ruta del directorio de nivel superior contiene espacios, entonces es necesario incluir toda la ruta entre comillas:

```
forfiles /s /p "C:\Documentos\Mis Partituras" /M *.ly /c "cmd /c lilypond @file"
```

Opciones básicas de la línea de órdenes para LilyPond

Están contempladas las siguientes opciones:

`-d`, `--define-default=`*variable*[`=`*valor*]
: Véase [Opciones avanzadas de línea de órdenes para LilyPond], página 6.

`-e`, `--evaluate=`*expresión*
: Evaluar la *expresión* de Scheme antes de analizar los archivos `.ly`. Se pueden pasar varias opciones `-e`, que se evalúan en secuencia.

  La expresión se evalúa en el módulo `guile-user`, de manera que si quiere usar una definición como `(define-public a 42)` como *expresión*, debe utilizar

  ```
  lilypond -e '(define-public a 42)'
  ```

  en la línea de órdenes, e incluir

  ```
  #(use-modules (guile-user))
  ```

  al principio del archivo `.ly`.

  > **Nota:** Los usuarios de Windows deben utilizar comillas dobles en lugar de apóstrofos simples.

`-E`, `--eps` Generar archivos EPS.

Esta opción equivale a establecer las opciones de la línea de órdenes de LilyPond a `--ps` y `-dlilypond-book-output`.

`-f`, `--format=`*formato*

Formato del archivo o archivos (principal/es) de salida. Los valores posibles de *formato* son `ps`, `pdf`, `png` o `svg`.

Ejemplo: `lilypond -fpng archivo.ly`

El SVG usa internamente un backend específico, y por tanto no se puede obtener dentro de la misma ejecución que otros formatos; el uso de `-fsvg` o de `--svg` son en realidad equivalentes al empleo de la opción `-dbackend=svg`. Véase [Opciones avanzadas de línea de órdenes para LilyPond], página 6.

`-h`, `--help`

Mostrar un resumen de las formas de utilización.

`-H`, `--header=`*CAMPO*

Volcar un campo de cabecera al archivo `NOMBREBASE.`*`CAMPO`*

A modo de ejemplo, vamos a suponer que tenemos un archivo de entrada `archivo.ly` que contiene

```
\header { title = "Título" }
\score { c1 }
```

La instrucción

```
lilypond -H title archivo.ly
```

crea entonces un archivo de texto sencillo `archivo.title` que contiene la cadena `Título`.

`-i`, `--init=`*archivo*

Establecer el archivo de inicio a *archivo* (predeterminado: `init.ly`).

`-I`, `--include=`*directorio*

Añadir el *directorio* a la ruta de búsqueda de archivos de entrada.

Se pueden escribir varias opciones `-I`. La búsqueda se inicia en el directorio actual, y no se encuentra el archivo de inclusión, la búsqueda continúa dentro del directorio especificado en la primera opción `-I`, a continuación en el directorio dado en la segunda opción `-I`, y así sucesivamente.

> **Nota:** El uso del carácter de tilde curva (`~`) con la opción `-I` puede producir resultados inesperados en algunos 'shells'.
>
> Los usuarios de Windows deben incluir una barra inclinada al final de la ruta del directorio.

`-j`, `--jail=`*usuario*`,`*grupo*`,`*jaula*`,`*directorio*

[Esta opción solo está disponible si su sistema operativo contempla la funcionalidad `chroot`. Concretamente, Windows no la contempla.]

Ejecutar `lilypond` en una jaula de chroot.

La opción `--jail` (jaula) proporciona una alternativa más flexible a la opción `-dsafe` cuando el proceso de tipografía de LilyPond está disponible a través de un servidor web o cuando LilyPond ejecuta instrucciones enviadas por fuentes externas (véase [Opciones avanzadas de línea de órdenes para LilyPond], página 6).

La opción `--jail` funciona cambiando la raíz de `lilypond` a *jaula* justo antes de comenzar el proceso de compilación en sí. Entonces se cambian el usuario y el grupo

a los que se han dado en la opción, y el directorio actual se cambia a *directorio*. Esta instalación garantiza que no es posible, al menos en teoría, escapar de la jaula. Observe que para que funcione `--jail`, se debe ejecutar `lilypond` como root, lo que normalmente se puede hacer de una forma segura utilizando `sudo`.

La instalación de una jaula puede ser un asunto relativamente complejo, pues debemos asegurarnos de que LilyPond puede encontrar *dentro* de la propia jaula todo lo que necesita para poder compilar la fuente. Una típica configuración de jaula de chroot consta de los siguientes elementos:

Preparar un sistema de archivos separado
: Se debe crear un sistema de archivos separado para LilyPond, de forma que se pueda montar con opciones seguras como `noexec`, `nodev` y `nosuid`. De esta forma, es imposible ejecutar programas o escribir directamente a un dispositivo desde LilyPond. Si no quiere crear una partición separada, tan sólo tiene que crear un archivo de un tamaño razonable y usarlo para montar un dispositivo loop. El sistema de archivos separado garantiza también que LilyPond nunca pueda escribir en un espacio mayor del que se le permita.

Preparar un usuario separado
: Se debe usar un usuario y grupo separados (digamos `lily/lily`) con bajos privilegios para ejecutar LilyPond dentro de la jaula. Debería existir un solo directorio con permisos de escritura para este usuario, y debe pasarse en el valor *directorio*.

Preparar la jaula
: LilyPond necesita leer algunos archivos mientras se ejecuta. Todos estos archivos se deben copiar dentro de la jaula, bajo la misma ruta en que aparecen en el sistema de archivos real de root. Todo el contenido de la instalación de LilyPond (por ejemplo `/usr/share/lilypond`) se debe copiar.

  Si surgen problemas, la forma más sencilla de rastrearlos es ejecutar LilyPond usando `strace`, lo que permite determinar qué archivos faltan.

Ejecutar LilyPond
: Dentro de una jaula montada con `noexec` es imposible ejecutar ningún programa externo. Por tanto, LilyPond se debe ejecutar con un backend que no necesite tal programa. Como ya hemos mencionado, se debe ejecutar con privilegios del superusuario (que por supuesto perderá inmediatamente), posiblemente usando `sudo`. También es una práctica recomendable limitar el número de segundos de tiempo de CPU que LilyPond puede usar (p.ej., usando `ulimit -t`), y, si su sistema operativo lo contempla, el tamaño de la memoria que se puede reservar. Véase también [LilyPond en una jaula de chroot], página 13.

`-l`, `--loglevel=`*NIVEL*
: Fijar el grado en que la salida de consola es prolija al nivel *NIVEL*. Los valores posibles son:

  `NONE` Ninguna salida en absoluto, ni siquiera mensajes de error.

  `ERROR` Solamente mensajes de error, no advertencias o indicaciones de progreso.

  `WARN` Advertencias y mensajes de error, no de progreso.

  `BASIC_PROGRESS`
  : Mensajes de progreso básicos (éxito), advertencias y errores.

`PROGRESS` Todos los mensajes de progreso, advertencias y errores.

`INFO` Mensajes de progreso, advertencias, errores e información de ejecución adicional. Este es el valor predeteminado.

`DEBUG` Todos los mensajes posibles, incuida la información de depuración prolija.

`-o`, `--output=`*archivo*
`-o`, `--output=`*carpeta*
Establecer el nombre del archivo de salida predeterminado a *archivo* o, si existe una carpeta con ese nombre, dirigir la salida hacia *carpeta*, tomando el nombre de archivo del documento de entrada. Se añade el sufijo correspondiente (por ejemplo, `.pdf` para PDF) en los dos casos.

`-O`, `--pspdfopt=`*clave*
Establecer la optimización de salida de PS o PDF a *clave*. Los valores posibles son:

`size` Generar un documento PS/EPS/PDF muy pequeño. Esta es la opción predeterminada.

La utilización de este valor equivale a fijar las opciones de la línea de órdenes del Scheme de LilyPond a `-dmusic-font-encodings='#f'` y `-dgs-never-embed-fonts='#f'`.

`TeX` Producir archivos que están optimizados para su inclusión dentro de documentos de pdfTEX, LuaTEX o XeTEX.

La utilización de este valor equivale a fijar las opciones de la línea de órdenes del Scheme de LilyPond a `-dmusic-font-encodings='#t'` y `-dgs-never-embed-fonts='#f'`.

`TeX-GS` Si queremos incluir más de un PDF generado por LilyPond dentro de un documento de TeX, utilice esta opción y post-procese el PDF generado por TeX con Ghostscript.

La utilización de este valor equivale a fijar las opciones de la línea de órdenes del Scheme de LilyPond a `-dmusic-font-encodings='#t'` y `-dgs-never-embed-fonts='#t'`.

–ps Generar PostScript.

`--png` Generar imágenes de las páginas en formato PNG. Esta opción equivale a `-fpng`.

La resolución de la imagen se puede fijar a *N* PPP con

```
-dresolution=N
```

`--pdf` Generar un PDF. Esta es la opción predeterminada, equivalente a `-fpdf`.

`-s`, `--silent`
No mostrar el avance, solo los mensajes de error. Equivale a `-lERROR`.

`--svg` Generar archivos SVG para cada página. Esta opción equivale a `-fsvg`.

`-v`, `--version`
Mostrar la información de la versión.

`-V`, `--verbose`
Ser prolijo: mostrar las rutas completas de todos los archivos que se leen, y dar información cronométrica.

`-w`, `--warranty`
Mostrar la garantía con que viene GNU LilyPond (¡no viene con **NINGUNA GARANTÍA**!).

Opciones avanzadas de línea de órdenes para LilyPond

La opción `-d` es la interfaz de la línea de órdenes a la función de Scheme de LilyPond `ly:set-option`. Esto significa que todas las opciones que se relacionan aquí pueden establecerse también dentro de los propios archivos `.ly`.

`-d, --define-default=`*opción*`[=`*valor*`]`
`-d, --define-default=no-`*opción*

Fijar el símbolo interno de Scheme equivalente *opción* a *valor*. Por ejemplo, la opción de línea de órdenes

```
-dbackend=svg
```

equivale a

```
#(ly:set-option 'backend 'svg)
```

dentro de un archivo de entrada de LilyPond.

Si no se proporciona el *valor*, usar `#t` como valor (lo que puede producir resultados extraños cuando el tipo esperado para el *valor* no es booleano). Se puede añadir el prefijo `no-` a la *opción* para 'desactivar' una opción, aportando `#f` como valor. Por ejemplo,

```
-dpoint-and-click='#f'
```

es lo mismo que

```
-dno-point-and-click
```

[Observe que el carácter '#' introduce un comentario en muchos shells. Por este motivo se recomienda siempre entrecomillar las expresiones que lo contengan.]

La siguiente tabla relaciona todos los nombres de opción junto con sus valores. Dentro del código de Scheme, los valores de opción se pueden leer usando la función `ly:get-option`.

`anti-alias-factor` *número*

Generar el gráfico a mayor resolución (usando el factor *número*) y reducir la escala para evitar el pixelado en las imágenes PNG. Predeterminado: `1.0`.

`aux-files` *bool*

Si *bool* es `#t`, crear archivos `.tex`, `.texi` y `.count` cuando se usa con la opción de backend `eps`. Predeterminado: `#t`.

`backend` *símbolo*

Usar *símbolo* como backend para la salida de LilyPond. Los valores posiboles son:

`ps` Es el ajuste predeterminado. Los archivos de PostScript incluyen las fuentes TTF, Type1 y OTF. No se genera ningún 'subconjunto' de dichas fuentes tipográficas. Advierta que el uso de conjuntos de caracteres 'orientales' como el japonés puede dar lugar a archivos de tamaño muy grande.

Para la salida PDF se utiliza también el backend `ps`; los datos PS resultantes se post-procesan mediante el guión `ps2pdf` de Ghostscript, que también efectúa subconjuntos de fuentes de manera predeterminada.

`svg` Gráficos vectoriales escalables. Por cada página de la salida, se crea un solo archivo `SVG`. Los glifos musicales se codifican como gráficos vectoriales, pero las fuentes tipográficas del texto *no* se incrustan en los archivos SVG. Cualquier visor de SVG necesita que las fuentes de texto correspondientes estén disponibles para la correcta representación tanto del texto como de la letra. Se recomienda no utilizar 'alias' ni 'listas' de fuentes tipográficas por si el visor de SVG no es capaz de manejarlas.

Al usar archivos de fuente abierta para la web *Web Open Font Format* (WOFF), es necesario indicar la opción `-dsvg-woff`

`clip-systems` *bool*
: Si *bool* es `#t`, extraer fragmentos de música de la partitura. Requiere que la función `clip-regions` esté definida dentro del bloque `\layout`. Véase Sección "Extracción de fragmentos de música" en *Referencia de la Notación*. No se extrae ningún fragmento si se usa con la opción `-dno-print-pages`. Predetermiadno: `#f`.

`crop` *bool* Si *bool* es `#t`, incorporar toda la música con sus títulos y encabezamientos, sin márgenes, en un archivo de salida de 'una sola página'. Predeterminado: `#f`.

`datadir` Prefijo de los archivos de datos. Esta opción es de solo lectura; su establecimiento no tiene ningún efecto.

`debug-skylines` *bool*
: Si *bool* es `#t`, efectuar una depuración de las líneas de horizonte. Predeterminado: `#f`.

`delete-intermediate-files` *bool*
: Si *bool* es `#t`, eliminar los archivos intermedios `.ps` inútiles que se crean durante la compilación. Predeterminado: `#t`.

`embed-source-code` *bool*
: Si *bool* es `#t`, empotrar los archivos de entrada en código de LilyPond dentro del documento PDF generado. Predeterminado: `#f`.

`eps-box-padding` *número*
: Rellenar el borde izquierdo de la caja contenedora (bounding box) del EPS de salida en *número* milímetros. Predeterminado: `#f` (que significa que no se rellena la caja contenedora).

`font-export-dir` *cadena*
: Fijar el directorio para la exportación de fuentes tipográficas como archivos de PostScript a *cadena*. Esto es de utilidad cuando queremos crear primero un PDF sin las fuentes incrustadas, e incrustar más tarde las fuentes con Ghostscript como se ve más abajo.

```
$ lilypond -dfont-export-dir=fontdir -dgs-never-embed-fonts foo.ly
$ gs -q -dBATCH -dNOPAUSE -sDEVICE=pdfwrite \
     -sOutputFile=foo.embedded.pdf foo.pdf fontdir/*.font.ps
```

Nota: a diferencia de `font-ps-resdir`, este método no puede empotrar las fuentes CID con Ghostscript 9.26 y posteriores.

Nota: de la misma manera que `font-ps-resdir`, esta opción se salta las fuentes TrueType porque el empotrado de fuentes TrueType produce un desbaratamiento de los caracteres. Para evitar este dessbaratabiento, use `gs-never-embed-fonts`, pues a pesar de su nombre esta opción empotra las fuentes TrueType.

Predeterminado: `#f` (que significa no exportar).

`font-ps-resdir` *cadena*
: Fijar el directorio (como *cadena*) para construir un subconjunto del directorio de recursos PostScript y usarlo para empotrar las fuentes más tarde. Esto es útil si queremos crear primero un PDF sin las fuentes empotradas y empotrar las fuentes más tarde con Ghostscript como se ve más abajo.

```
$ lilypond -dfont-ps-resdir=resdir -dgs-never-embed-fonts foo.ly
$ gs -q -dBATCH -dNOPAUSE -sDEVICE=pdfwrite \
     -I resdir -I resdir/Font \
```

```
-sOutputFile=foo.embedded.pdf foo.pdf
```

Nota: es mejor no especificar el directorio que contiene el nombre `Resource` porque este tiene un significado especial al especificarlo con la opción `-I` para Ghostscript.

Nota: a diferencia de `font-export-dir`, este método puede empotrar fuentes CID con Ghostscript 9.26 y posteriores.

Nota: de igual manera que `font-export-dir`, esta opción se salta las fuentes TrueType porque el empotrado tardío de fuentes TrueType causa un desbaratamiento de los caracteres. Para evitar el desbaratamiento, use `gs-never-embed-fonts`, ya que este empotra las fuentes TrueType a pesar de su nombre.

Predeterminado: `#f` (que significa no construir).

`gs-load-fonts` *bool*
: Si *bool* es `#t`, cargar fuentes a través de Ghostscript. Esta opción hace que los archivos de salida de LilyPond solo contenga referencias a todas las fuentes, lo que debe después resolverse a las fuentes reales en un paso de post-procesado por medio de Ghostscript. Predeterminado: `#f`.

`gs-load-lily-fonts` *bool*
: Si *bool* es `#t`, cargar las fuentes de LilyPond por medio de Ghostscript. Esta opción hace que los archivos de salida de LilyPond contengan solamente referencias a sus fuentes de música, lo que debe después resolverse a las fuentes reales en un paso de post-procesado por medio de Ghostscript. Todas las demás fuentes se dirigen a la salida como es habitual. Predeterminado: `#f`.

`gs-never-embed-fonts` *bool*
: Si *bool* es `#t`, hacer que Ghostscript incruste solamente tipos de letra TrueType y ningún otro formato de fuente. Predeterminado: `#f`.

`help` *bool* Si *bool* es `#t`, mostrar esta ayuda. Predeterminado: `#f`.

`include-book-title-preview` *bool*
: Si *bool* es `#t`, incluir los títulos de libro en las imágenes de vista previa. Predeterminado: `#t`.

`include-eps-fonts` *bool*
: Si *bool* es `#t`, incluir las fuentes tipográficas en los archivos EPS de cada uno de los sistemas. Predeterminado: `#t`.

`include-settings` *cadena*
: Incluir el archivo *cadena* de ajustes globales, que se incluye antes de que la partitura se procese Predeterminado: `#f` (que significa que no se incluye ningún archivo de ajustes globales).

`job-count` *número*
: Procesar en paralelo, utilizando *número* tareas. Predeterminado: `#f` (que significa que no se hace ningún procesado paralelo).

`log-file` *cadena*
: Redirigir la salida hacia el archivo de registro *cadena*`.log`. Predeterminado: `#f` (que significa ningún archivo de registro).

`max-markup-depth` *número*
: Fijar la profundidad máxima del árbol de marcado al valor *número*. Si un elemento de marcado tiene más niveles, suponer que no va a finalizar por sí mismo, imprimir un mensaje de advertencia y devolver un elemento de marcado nulo en su lugar. Predeterminado: `1024`.

`midi-extension` *cadena*
: Fijar la extensión de archivo predeterminada para los archivos MIDI de salida a *.cadena*. Predeterminado: `"midi"`.

`music-strings-to-paths` *bool*
: Si *bool* es `#t`, converir las cadenas de texto a trayectos cuando los glifos pertenecen a una fuente musical. Predeterminado: `#f`.

`paper-size` *extra-quoted-string*
: Establecer el tamaño predeterminado de papel a *extra-quoted-string*. Observe que la cadena se debe encerrar dentro de comillas con prefijo de escape. Predeterminado: `"\"a4\""`.

`pixmap-format` *symbol*
: Establecer el formato de salida de Ghostscript para las imágenes de matriz de puntos a *symbol*. Predeterminado: `png16m`.

`point-and-click` *bool*
: Si *bool* es `#t`, añadir enlaces de 'apuntar y pulsar' links a las salidas de PDF y SVG. Véase Sección 4.1 [Apuntar y pulsar], página 44. Predeterminado: `#t`.

`preview` *bool*
: Si *bool* es `#t`, crear imágenes de vista previa además de la salida normal. Predeterminado: `#f`.

  Esta opción está contemplada por todos los 'back-ends' (`pdf`, `png`, `ps`, `eps` y `svg`), excepto `scm`. Para un nombre de archivo de entrada *archivo* y un backend *formato*, genera un archivo de salida, con el nombre *archivo*`.preview.`*formato*, que contiene los títulos y el primer sistema de la música. Si se están utilizando bloques `\book` o `\bookpart`, aparecen en la salida los títulos de `\book`, `\bookpart` o `\score`, incluido el primer sistema de cada bloque `\score` si la variable de `\paper` `print-all-headers` está fijada al valor `#t`.

  Para suprimir la salida usual, utilice las opciones `-dprint-pages` o `-dno-print-pages` según sus necesidades.

`print-pages` *bool*
: Si *bool* es `#t`, generar páginas completas. Predeterminado: `#t`.

  La opción `-dno-print-pages` es útil en combinación con las opciones `-dpreview` o `-dcrop`.

`protected-scheme-parsing` *bool*
: Si *bool* es `#t`, continuar cuando se interceptan errores en el código empotrado de Scheme dentro del analizador sintáctico. Si se establece al valor `#f`, detenerse cuando se produzca algún error e imprimir una traza de la pila. Predeterminado: `#t`.

`relative-includes` *bool*
: Cuando se procesa una instrucción `\include`, buscar el archivo de inclusión en una ruta relativa al archivo actual si *bool* es `#t`. Si se establece a `#f`, buscar el archivo según una ruta relativa al archivo raíz. Predeterminado: `#t`.

`resolution` *número*
: Fijar la resolución para generar imágenes de matriz de puntos `PNG` a *número* ppp. Predeterminado: `101`.

`safe` *#f* No confiar en la entrada `.ly`.

  Cuando el servicio de tipografía está disponible a través de un servidor web, **SE DEBEN** pasar las opciones `-dsafe` o `--jail`. La opción `-dsafe` evita que el código de Scheme empotrado pueda producir un desastre, p.ej.:

```
% demasiado peligroso para escribirlo correctamente
```

```
#(s ystem "rm -rf /")
% malicioso pero no destructivo
{ c4^$(ly:gulp-file "/etc/passwd") }
```

La opción `-dsafe` funciona evaluando las expresiones de Scheme en línea dentro de un módulo seguro especial. Deriva del módulo `safe-r5rs` de GUILE, pero además añade unas cuantas funciones de la API de LilyPond que están relacionadas en `scm/safe-lily.scm`.

Además, el modo seguro prohíbe las directivas `\include` y desactiva la utilización de barras invertidas en las cadenas de TEX. Asimismo, no es posible importar variables de LilyPond dentro de Scheme cuando se está en modo seguro.

La opción `-dsafe` *no* detecta la sobreutilización de recursos, por lo que aún es posible hacer que el programa se cuelgue indefinidamente, por ejemplo suministrando estructuras de datos cíclicas en el backend. Por ello, si está usando LilyPond en un servidor web accesible públicamente, el proceso se debe limitar tanto en el uso de memoria como de CPU.

El modo seguro evita que se puedan compilar muchos fragmentos de código útiles.

La opción `--jail` es una alternativa más segura aún, pero requiere más trabajo para su configuración. Véase [Opciones básicas de la línea de órdenes para LilyPond], página 2.

`separate-log-files` *bool*
: Para los archivos de entrada `archivo1.ly`, `archivo2.ly`, ..., dar salida a datos de registro hacia los archivos `archivo1.log`, `archivo2.log`, ..., si *bool* es `#t`. Predeterminado: `#f`.

`show-available-fonts` *bool*
: Si *bool* es `#t`, imprimir un listado de los nombres de fuente tipográfica disponibles tal y como los proporciona la biblioteca fontconfig. Al final de esta lista, LilyPond presenta los ajustes de configuración del propio fontconfig. Predeterminado: `#f`.

`strip-output-dir` *bool*
: Si *bool* es `#t`, no utilizar la parte del directorio tomada de las rutas de los archivos cuando se construyen nombres de archivos de salida. Predeterminado: `#t`.

`strokeadjust` *bool*
: Si *bool* es `#t`, forzar el ajuste de los trazos de PostScript. Esta opción es relevante principalmente cuando se genera un PDF a partir de la salida de PostScript (el ajuste del trazo está por lo general activado automáticamente para dispositivos de mapa de puntos de baja resolución). Sin esta opción, los visores de PDF tienden a producir anchuras de plica muy poco consistentes con las resoluciones típicas de las pantallas de ordenador. Sin embargo, la opción no afecta de forma muy significativa a la calidad de la impresión y causa grandes incrementos en el tamaño del archivo PDF. Predeterminado: `#f`.

`svg-woff` *bool*
: Esta opción es necesaria al usar archivos de fuente abierta para la Web, Web Open Font Format (WOFF) con el backend `svg`. Si *bool* es `#t`, se crea un solo archivo SVG para cada página de salida. Aparte de los glifos musicales propios de LilyPond, no se incluye ninguna otra información de fuente tipográfica. Todo visor de SVG necesita, por ello, tener las fuentes disponibles para la representación correcta tanto del texto como de la letra. Asimismo se recomienda no usar alias de fuentes ni listas, por si el visor de SVG no es capaz de manejarlos. Predeterminado: `#f`.

`verbose`
: Nivel de verbosidad. Esta es una opción de solo lectura; su establecimiento no tiene ningún efecto.

`warning-as-error` *bool*
: Si *bool* es `#t`, cambiar todos los mensajes de advertencia y de 'error de programación' a errores. Predeterminado: `#f`.

Variables de entorno

`lilypond` reconoce las siguientes variables de entorno:

`LILYPOND_DATADIR`
: Especifica un directorio en el que los mensajes de localización y de datos se buscan de forma predeterminada, sobreescribiendo las ubicaciones definidas en tiempo de compilación o computadas dinámicamente en tiempo de ejecución (véase [Reubicación del programa, página 11). El directorio debe contener subdirectorios llamados `ly`, `ps`, `tex`, etc.

`LILYPOND_LOCALEDIR`
: Espcificar el directorio en que están los archivos específicos de localización. Sobreescribe el valor derivado de `LILYPOND_DATADIR`.

`LILYPOND_RELOCDIR`
: Especificar el directorio en que se ubican los archivos de relocalización. Sobreescribe el valor que se deriva de la ubicación del binario `lilypond`.

`LANG`
: Idioma de los datos de LilyPond enviados a `stdout` y a `stderr`, por ejemplo informes del avance, mensajes de advertencia o salida de depuración. Ejemplo: `LANG=es`.

`LILYPOND_LOGLEVEL`
: Nivel de registro predeterminado. Si LilyPond se llama sin ningún nivel de registro explícito (es decir, sin opción de línea de órdenes `--loglevel`), se usa este valor.

`LILYPOND_GC_YIELD`
: Una variable, como porcentaje, que ajusta el comportamiento de la administración de memoria. Con valores más altos, el programa usa más memoria; con valores más bajos, usa más tiempo de CPU. El valor predeterminado es `70`.

`TMPDIR`
: Esto especifica el directorio temporal en GNU/Linux y Mac. EL predeterminado es `/tmp`. Es el directorio en que se guardan los archivos intermedios (tales como archivos de PostScript) durante la compilación. La sobreescritura de esta variable podría ser útil, por ejemplo, si el usuario que ejecuta lilypond no tiene privilegios de escritura sobre el directorio temporal predeteminado. Ejemplo: `TMPDIR=~/foo`.

Reubicación del programa

Casi todos los programas dentro del mundo Unix utilizan directorios predeterminados para sus datos, que vienen determinados en el momento de la configuración y antes de la compilación. LilyPond no es una excepción; por ejemplo, una instalación típica coloca el binario `lilypond` en `/usr/bin` y todos los archivos específicos de LilyPond dentro de subdirectorios de `/usr/share/lilypond/2.21.0/` (suponiendo que la versión actual es 2.21.0).

Aunque este enfoque funciona perfectamente para la compilación manual y para plataformas que vienen con gestores de paquetes estandarizados, puede producir algunos problemas allí donde los mencionados gestores no son tan comunes o no se utilizan por defecto. Son ejemplos típicos de este tipo de plataformas Windows y MacOS, donde los usuarios dan por sentado que todo lo que viene con la aplicación se puede ubicar en cualquier lugar.

La solución habitual a este problema es que el programa contemple la reubicación: en lugar de usar rutas fijas a los archivos de datos, las ubicaciones a los necesarios archivos de apoyo se calculan en tiempo de ejecución *relativos al binario que se ejecuta*.

Archivos de reubicación

En realidad existe un segundo mecanismo para la configuración en tiempo de ejecución: LilyPond se apoya fuertemente en bibliotecas y programas externos, especialmente las bibliotecas 'FontConfig' y 'GUILE' para encontrar las fuentes tipográficas del sistema y manejar los archivos de Scheme, respectivamente, y el programa `gs` (Ghostscript) para convertir datos de PS a archivos PDF. Todos ellos se deben configurar también para tener localizados sus respectivos archivos de datos. Para ello, el programa `lilypond` analiza todos los archivos que están dentro de un directorio llamado `relocate` (si existe; véase más abajo dónde se busca este directorio) para manipular las variables de entorno, las que a su vez controlan estas bibliotecas y programas externos. El formato de estos archivos de reubicación es sencillo; cada línea tiene la sintaxis siguiente:

instrucción clave=valor

y las líneas vacías se ignoran.

La directiva *instrucción* es una de las siguientes.

`set` Fijar incondicionalmente la variable *clave* a *valor*. Esto sobreescribe cualquier valor establecido previamente.

`set?` Fijar la variable de entorno *clave* a *valor* solamente si *clave* no está definida aún. Dicho de otra forma, no sobreescribe ningún valor fijado previamente.

`setdir` Si *valor* es un directorio, fijar incondicionalmente la variable de entorno *clave* a *valor*. En caso contrario, emitir un mensaje de advertencia.

`setfile` Si *valor* es un archivo, fijar incondicionalmente la variable de entorno *clave* a *valor*. En caso contrario, emitir un mensaje de advertencia.

`prependdir`
Anadir el nombre del directorio *valor* al principio de la lista de directorios de la variable de entorno *clave*. Si *clave* no existe, se crea.

Se permite usar variables de entorno (prefijadas por el símbolo del dólar) dentro de *valor* y se expanden antes de que la directiva se ejecute.

He aquí dos ejemplos de entradas del archivo de reubicación.

```
set? FONTCONFIG_FILE=$INSTALLER_PREFIX/etc/fonts/fonts.conf
prependdir GUILE_LOAD_PATH=$INSTALLER_PREFIX/share/guile/1.8
```

Se debe evitar que más de una línea establezca el valor de la misma variable de entorno dentro de los archivos de reubicación, ya que el orden de la exploración de los archivos en el directorio `relocate` es arbitrario.

Algoritmo de la reubicación

LilyPond usa el siguiente algoritmo para buscar los archivos de datos.

1. Calcular el directorio en que se encuentra el archivo binario `lilypond` que se está ejecutando actualmente. Le llamaremos `bindir`. Fijar la variable de entorno (itnerna) `INSTALLER_PREFIX` a *`bindir`*`/..` (esto es, el directorio padre de `bindir`).
2. Comprobar la variable de entorno `LILYPOND_DATADIR`. Si está establecida, usar su valor como el directorio de datos de LilyPond, `datadir`. En caso contrario, usar o bien `$INSTALLER_PREFIX/share/lilypond/`*`versión`* (siendo *versión* la versión actual de LilyPond) o bien `$INSTALLER_PREFIX/share/lilypond/current`.
3. Comprobar la variable de entorno `LILYPOND_LOCALEDIR`. Si está establecida, usar su valor como la carpeta de datos de localización internacional de LilyPond, `localedir`. En caso contrario, usar `$INSTALLER_PREFIX/share/locale`.

4. Comprobar la variable de entorno `LILYPOND_RELOCDIR`. Si está establecida, usar su valor como el directorio de los archivos de reubicación de LilyPond, `relocdir`. En caso contrario, usar `$INSTALLER_PREFIX/etc/relocate`.
5. Si `datadir` no existe, usar en su lugar un valor calculado en tiempo de compliación. Lo mismo para `localedir` (pero no para `relocdir`, puesto que no tiene razón de ser).
6. Si `relocdir` existe, procesar todos los archivos de este directorio como se describe en [Archivos de reubicación], página 12.

LilyPond en una jaula de chroot

La preparación del servidor para que ejecute LilyPond en una jaula de chroot es una tarea muy complicada. Los pasos están relacionados más abajo. Los ejemplos que aparecen en cada uno de los pasos son válidos para Ubuntu GNU/Linux, y pueden requerir el uso de `sudo` según corresponda.

- Instale los paquetes necesarios: LilyPond, Ghostscript e ImageMagick.
- Cree un usuario nuevo con el nombre de `lily`:

  ```
  adduser lily
  ```

  Esto también creará un nuevo grupo para el usuario `lily`, y una carpeta personal, `/home/lily`
- En la carpeta personal del usuario `lily`, cree un archivo para usarlo como un sistema de archivos separado:

  ```
  dd if=/dev/zero of=/home/lily/loopfile bs=1k count= 200000
  ```

  Este ejemplo crea un archivo de 200MB para su uso como el sistema de archivos de la jaula.
- Cree un dispositivo loop, haga un sistema de archivos y móntelo, después cree una carpeta que sea escribible por el usuario `lily`:

  ```
  mkdir /mnt/lilyloop
  losetup /dev/loop0 /home/lily/loopfile
  mkfs -t ext3 /dev/loop0 200000
  mount -t ext3 /dev/loop0 /mnt/lilyloop
  mkdir /mnt/lilyloop/lilyhome
  chown lily /mnt/lilyloop/lilyhome
  ```
- En la configuración de los servidores, JAIL será `/mnt/lilyloop` y DIR será `/lilyhome`.
- Cree un gran árbol de directorios dentro de la jaula copiando los archivos necesarios, como se muestra en el guión de ejemplo que aparece más abajo.

  Puede usar `sed` para crear los archivos de copia necesarios para un ejecutable dado:

  ```
  for i in "/usr/local/lilypond/usr/bin/lilypond" "/bin/sh" "/usr/bin/;  \
    do ldd $i | sed 's/.*=> \/\(.*\/\)\([^(]*\).*/mkdir -p \1 \&\&  \
      cp -L \/\1\2 \1\2/' | sed 's/\t\/\(.*\/\)\(.*\) (.*)$/mkdir -p  \
        \1 \&\& cp -L \/\1\2 \1\2/' | sed '/.*=>.*/d'; done
  ```

Guión de ejemplo para Ubuntu 8.04 de 32 bits

```
#!/bin/sh
## aquí se fijan los valores predeterminados

username=lily
home=/home
loopdevice=/dev/loop0
jaildir=/mnt/lilyloop
# prefijo (¡sin la barra inicial!)
```

```
lilyprefix=usr/local
# el directorio en que lilypond se encuentra instalado en el sistema
lilydir=/$lilyprefix/lilypond/

userhome=$home/$username
loopfile=$userhome/loopfile
adduser $username
dd if=/dev/zero of=$loopfile bs=1k count=200000
mkdir $jaildir
losetup $loopdevice $loopfile
mkfs -t ext3 $loopdevice 200000
mount -t ext3 $loopdevice $jaildir
mkdir $jaildir/lilyhome
chown $username $jaildir/lilyhome
cd $jaildir

mkdir -p bin usr/bin usr/share usr/lib usr/share/fonts $lilyprefix tmp
chmod a+w tmp

cp -r -L $lilydir $lilyprefix
cp -L /bin/sh /bin/rm bin
cp -L /usr/bin/convert /usr/bin/gs usr/bin
cp -L /usr/share/fonts/truetype usr/share/fonts

# Ahora la magia de copiar las bibliotecas
for i in "$lilydir/usr/bin/lilypond" "$lilydir/usr/bin/guile" "/bin/sh"  \
  "/bin/rm" "/usr/bin/gs" "/usr/bin/convert"; do ldd $i | sed 's/.*=>  \
    \/\(.*\/\)\([^(]*\).*/mkdir -p \1 \&\& cp -L \/\1\2 \1\2/' | sed  \
      's/\t\/\(.*\/\)\(.*\) (.*)$/mkdir -p \1 \&\& cp -L \/\1\2 \1\2/'  \
        | sed '/.*=>.*/d'; done | sh -s

# Los archivos compartidos para Ghostscript...
      cp -L -r /usr/share/ghostscript usr/share
# Los archivos compartidos para ImageMagick
      cp -L -r /usr/lib/ImageMagick* usr/lib

### Ahora, suponiendo que tenemos test.ly en /mnt/lilyloop/lilyhome,
### deberíamos poder ejecutar:
### Observe que /$lilyprefix/bin/lilypond es un guión, que establece
### un valor para LD_LIBRARY_PATH : esto es crucial
      /$lilyprefix/bin/lilypond -jlily,lily,/mnt/lilyloop,/lilyhome test.ly
```

1.3 Mensajes de error

Pueden aparecer distintos mensajes de error al compilar un archivo:

Advertencia
: Algo tiene un aspecto sospechoso. Si estamos pidiendo algo fuera de lo común, entenderemos el mensaje y podremos ignorarlo. Sin embargo, las advertencias suelen indicar que algo va mal con el archivo de entrada.

Error
: Algo va claramente mal. El paso actual de procesamiento (análisis, interpretación o formateo visual) se dará por terminado, pero el siguiente paso se saltará.

Error fatal
: Algo va claramente mal, y LilyPond no puede seguir. Rara vez sucede esto. La causa más frecuente son las tipografías mal instaladas.

Error de Scheme
: Los errores que ocurren al ejecutar código de Scheme se interceptan por parte del intérprete de Scheme. Si se está ejecutando con las opciones `-V` o `--verbose` (prolijo) entonces se imprime una traza de llamadas de la función ofensiva.

Error de programación
: Ha habido algún tipo de inconsistencia interna. Estos mensajes de error están orientados a ayudar a los programadores y a los depuradores. Normalmente se pueden ignorar. En ocasiones aparecen en cantidades tan grandes que pueden entorpecer la visión de otros mensajes de salida.

Abortado (volcado de core)
: Esto señala un error de programación serio que ha causado la interrupción abrupta del programa. Estos errores se consideran críticos. Si se topa con uno, envíe un informe de fallo.

Si los errores y advertencias se pueden ligar a un punto del archivo de entrada, los mensajes tienen la forma siguiente:

```
archivo:línea:columna: mensaje
línea de entrada problemática
```

Se inserta un salto de línea en la línea problemática para indicar la columna en que se encontró el error. Por ejemplo,

```
prueba.ly:2:19: error: no es una duración: 5
  { c'4 e'
           5 g' }
```

Estas posiciones son la mejor suposición de LilyPond sobre dónde se ha producido el mensaje de error, pero (por su propia naturaleza) las advertencias y errores se producen cuando ocurre algo inesperado. Si no ve un error en la línea que se indica del archivo de entrada, trate de comprobar una o dos líneas por encima de la posición indicada.

Observe que los diagnósticos se pueden activar en cualquier punto durante las numerosas fases del procesado. Por ejemplo, si hay partes de la entrada que se procesan varias veces (como en la salida midi y de disposición de la página), o si la misma variable musical se utiliza en más de un contexto, puede aparecer el mismo mensaje varias veces. Los diagnósticos producidos en un estado 'tardío' (como las comprobaciones de compás) también podrían emitirse más de una vez.

Se ofrece más información sobre los errores en la sección Sección 1.4 [Errores comunes], página 15.

1.4 Errores comunes

Las condiciones de error que se describen más abajo se producen con frecuencia, aunque su causa no es obvia o fácil de encontrar. Una vez se han visto y comprendido, se manejan sin problema.

La música se sale de la página

La música que se sale de la página por el margen derecho o que aparece exageradamente comprimida está causada casi siempre por haber introducido una duración incorrecta para una nota, produciendo que la nota final de un compás se extienda más allá de la línea divisoria. Esto no es inválido si la nota final de un compás no termina sobre la línea divisoria introducida automáticamente, pues simplemente se supone que la nota se solapa encima del siguiente compás.

Pero si se produce una larga secuencia tales notas solapadas, la música puede aparecer comprimida o salirse de la página porque los saltos de línea automáticos solamente se pueden insertar al final de compases completos, es decir, aquellos en que todas las notas terminan antes de o justo al final del compás.

> **Nota:** Una duración incorrecta puede hacer que se inhiban los saltos de línea, lo que llevaría a una sola línea de música muy comprimida o que se salga de la página.

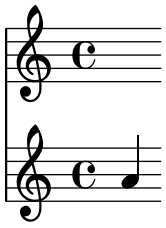
La duración incorrecta se puede encontrar fácilmente si se utilizan comprobaciones de compás, véase Sección "Comprobación de compás y de número de compás" en *Referencia de la Notación*.

Si realmente queremos tener una serie de estos compases con notas solapadas, debemos insertar una línea divisoria invisible donde queramos el salto de línea. Para ver más detalles, consulte Sección "Barras de compás" en *Referencia de la Notación*.

Aparece un pentagrama de más

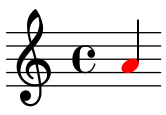
Si no se crean los contextos explícitamente con `\new` o con `\context`, se crearán discretamente tan pronto como se encuentra una instrucción que no se puede aplicar a un contexto existente. En partituras sencillas, la creación automática de los contextos es útil, y casi todos los ejemplos de los manuales de LilyPond se aprovechan de esta simplificación. Pero ocasionalmente la creación discreta de contextos puede hacer aflorar pentagramas o partituras nuevos e inesperados. Por ejemplo, podría esperarse que el código siguiente hiciera que todas las notas dentro del pentagrama siguiente estuvieran coloreadas de rojo, pero de hecho el resultado son dos pentagramas, permaneciendo el de abajo con las notas en el color negro predeterminado.

```
\override Staff.NoteHead.color = #red
\new Staff { a' }
```



Esto es así porque no existe ningún contexto `Staff` cuando se procesa la instrucción override de sobreescritura, se crea uno implícitamente y la sobreescriturase aplica a éste, pero entonces la instrucción `\new Staff` crea un pentagrama nuevo y distinto, en el que se colocan las notas. El código correcto para colorear todas las notas de rojo es

```
\new Staff {
  \override Staff.NoteHead.color = #red
  a'
}
```



Mensaje de error Unbound variable %

Este mensaje de error aparece al final de los mensajes de la consola o del archivo de registro junto a un mensaje "GUILE señaló un error . . ." cada vez que se llame a una rutina de Scheme que (incorrectamente) contenga un comentario *de LilyPond* en lugar de un comentario *de Scheme*.

Los comentarios de LilyPond comienzan con un símbolo de porcentaje, (%), y no se deben utilizar dentro de las rutinas de Scheme. Los comentarios de Scheme comienzan con punto y coma, (;).

Mensaje de error FT_Get_Glyph_Name

Este mensaje de error aparece en la salida de la consola o en el archivo log de registro si un archivo de entrada contiene un carácter que no es ASCII y no se ha guardado en la codificación de caracteres UTF-8. Para ver más detalles, consulte Sección "Codificación del texto" en *Referencia de la Notación*.

Advertencia sobre que las afinidades del pentagrama sólo deben decrecer

Esta advertencia puede aparecer si no hay ningún pentagrama en la salida impresa, por ejemplo si sólo hay un contexto `ChordName` y un contexto `Lyrics` como en una hoja guía de acordes. Los mensajes de advertencia se pueden evitar haciendo que uno de los contextos se comporte como un pentagrama, insertando

```
\override VerticalAxisGroup.staff-affinity = ##f
```

al comienzo. Para ver más detalles, consulte "Espaciado de las líneas que no son pautas" en Sección "Espaciado vertical flexible dentro de los sistemas" en *Referencia de la Notación*.

Mensaje de error Unexpected \new

Un bloque `\score` debe contener una *única* expresión musical. Si en vez de ello contiene varias instrucciones `\new Staff`, `\new StaffGroup` o contextos similares introducidos con `\new` sin que se hayan encerrado entre llaves curvas, `{ ... }`, o dobles paréntesis en ángulo, `<< ... >>`, así:

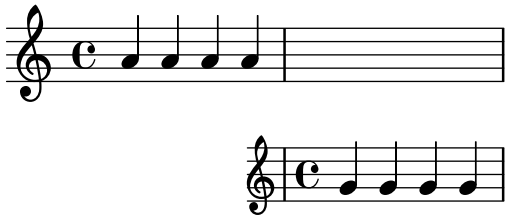
```
\score {
  % Invalid! Generates error: syntax error, unexpected \new
  \new Staff { ... }
  \new Staff { ... }
}
```

entonces se producirá el mensaje de error.

Para evitar el error, encierre todas las instrucciones `\new` dentro de llaves curvas o dobles paréntesis en ángulo.

El uso de llaves curvas introduce las instrucciones `\new` de forma secuencial:

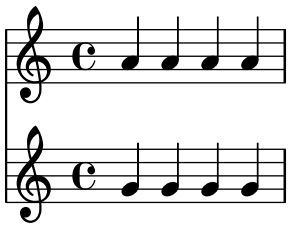
```
\score {
  {
    \new Staff { a' a' a' a' }
    \new Staff { g' g' g' g' }
  }
}
```



pero es más probable que se encuentre utilizando dobles ángulos de manera que los pentagramas nuevos se inserten en paralelo, es decir, simultáneamente:

```
\score {
```

```
  <<
    \new Staff { a' a' a' a' }
    \new Staff { g' g' g' g' }
  >>
}
```



Advertencia de que una voz requiere ajustes `\voiceXx` o `\shiftXx`

Si acontecen dos notas de distintas voces son plicas en la misma dirección y en el mismo momento musical, pero las voces no tienen especificado ningún desplazamiento de voz específico, aparece el mensaje 'advertencia: esta voz requiere ajustes \voiceXx o \shiftXx' cuando se compila el archivo de LilyPond. La advertencia aparece incluso si las notas no tienen plicas visibres, por ejemplo redondas, si las plicas de figuras más breves que tuviesen las misma altura, estuvieran en la misma dirección.

Recuerde que la dirección de la plica depende de la posición de la nota sobre el pentagrama a no ser que la dirección de la plica venga especificada, por ejemplo mediante `\voiceOne`. En este caso, la advertencia aparece solamente cuando las plicas están en la misma dirección, es decir, cuando se encuentran en la misma mitad del pentagrama.

Situando las notas en voces que tengan direcciones de plica y desplazamientos especificados, por ejemplo usando `\voiceOne` y otras instrucciones, se pueden evitar estos mensajes.

Las notas que están en voces con numeración más alta, `\voiceThree` y siguientes, se desplazan automáticamente para evitar las colisiones entre columnas. Esto produce un desplazamiento que es visible en las notas con plica, pero las redondas no se desplazan de forma visible a no ser que se produzca una verdadera colisión, o si las voces se cruzan respecto a su orden natural (cuando una voz con `\voiceThree` es más aguda que otra con `\voiceOne`, y casos semejantes).

Véase también

Sección "Voces explícitas" en *Manual de Aprendizaje*, Sección "Ejemplos reales de música" en *Manual de Aprendizaje*, Sección "Polifonía en un solo pentagrama" en *Referencia de la Notación*, Sección "Resolución de las colisiones" en *Referencia de la Notación*.

2 Actualizar ficheros con `convert-ly`

Según LilyPond va mejorando, puede cambiar la sintaxis de algunas instrucciones y funcioens del lenguaje de entrada. Ello puede conducir a errores inesperados, advertencias y hasta salida errónea cada vez que se utilizan con la versión actual de LilyPond archivos de entrada que habían sido creados para versiones anteriores.

Como ayuda para este problema, puede usarse la herramienta `convert-ly` para actualizar esos archivos de entrada antiguos y que sigan la sintaxis nueva.

2.1 ¿Por qué cambia la sintaxis?

Con frecuencia, los cambios en la sintaxis se llevan a cabo para hacer que la entrada sea más sencilla tanto de leer como de escribir, pero en ocasiones se hacen los cambios para acomodar nuevas funcionalidades o mejoras para las funciones existentes.

Lo ilustramos a continuación con un ejemplo real:

Se supone que todos los nombres de las propiedades de `\paper` y de `\layout` están escritos en la forma `primero-segundo-tercero`. Sin embargo, en la versión 2.11.60, observamos que la propiedad `printallheaders` no seguía esta convención. ¿Deberíamos dejarla como está (confundiendo a los nuevos usuarios que tienen que tratar con un formato de entrada inconsistente), o cambiarla (fastidiando a los usuarios con experiencia que tienen partituras antiguas)?

Se tomó la decisión de cambiar el nombre de la propiedad por `print-all-headers`, y mediante el uso de la herramienta `convert-ly` se dio a los usuarios existentes la posibiilidad de actualizar automáticamente los archivos de entrada que tenían previamente.

Sin embargo, el uso de la herramienta `convert-ly` no permite tratar todos los cambios de sintaxis. En versiones de LilyPond anteriores a la 2.4.2, los acentos y las letras no inglesas se introducían utilizando LaTeX: por ejemplo, `No\"el` (que significa 'Navidad' en francés). Pero a partir de LilyPond 2.6, el carácter especial ë debe introducirse directamente en el archivo de LilyPond como un carácter UTF-8. La herramienta `convert-ly` no sabe cómo cambiar los caracteres especiales de LaTeX a caracteres de UTF-8; tendrá que actualizar manualmente sus archivos de LilyPond antiguos.

Las reglas de conversión de `convert-ly` funcionan usando correspondencia y sustitución de patrones de texto en lugar de una 'comprensión' profunda de los cambios producidos en un archivo dado. Esto tiene varias consecuencias:

- El buen funcionamiento de la conversión depende de la calidad de cada conjunto de reglas que se aplican y de la complejidad del cambio correspondiente. A veces las conversiones pueden necesitar correcciones manuales adicionales, por lo que los archivos originales deberían conservarse a efectos de comparación, si es necesario.
- Solamente son posibles las conversiones a las sintaxis más recientes: no existe ningún conjunto de reglas para volver a versiones más antiguas de LilyPond. Así pues, el archivo de entrada solamente se debe actualizar cuando ya no se mantienen las versiones antiguas de LilyPond. De nuevo, es conveniente conservar, por si acaso, los archivos de entrada, quizá mediante el uso de un sistema de control de versiones como el Git, que puede ser de gran ayuda para realizar el mantenimiento de varias versiones de los mismos archivos.
- LilyPond es bastante robusto al procesar espacios añadidos y suprimidos de manera "creativa", pero las reglas utilizadas por `convert-ly` con frecuencia hacen ciertas suposiciones de estilo. Por tanto, se recomienda seguir el estilo de la entrada tal y como se usa en los manuales de LilyPond para que las actualizaciones sean indoloras, especialmente porque todos los ejemplos de los propios manuales se actualizan usando la herramienta `convert-ly`.

2.2 Invocar convert-ly

La herramienta convert-ly utiliza los enunciados \version del archivo de entrada para detectar el número de versión antiguo. En casi todos los casos, para actualizar el archivo de entrada basta con ejecutar lo siguiente:

```
convert-ly -e miarchivo.ly
```

dentro del directorio que contiene el archivo de entrada. Con esto se actualiza miarchivo.ly *in situ* y se preserva el archivo original renombrándolo como miarchivo.ly~. Se modifica también el número de \version en el archivo actualizado además de la necesaria puesta al día de la sintaxis.

Al ejecutarse, la herramienta convert-ly imprime los números de versión de las conversiones que se han hecho. Si no aparece en el listado ningún número de versión para este archivo, significa que ya está actualizado y que es compatible con la sintaxis de la última versión de LilyPond.

> **Nota:** Para cada versión nueva de LilyPond, se crea una herramienta convert-ly asimismo nueva, aunque no todas y cada una de las versiones de LilyPond requiere cambios en la sintaxis de sus archivos de entrada a partir de la versión anterior. Ello significa que la herramienta convert-ly solamente convierte archivos hasta el último cambio de sintaxis que tiene, lo que a su vez podría implicar que el número de \version que se escribe en el archivo actualizado es, a veces, anterior que la versión de la propia herramienta convert-ly.

Para convertir todos los archivos de entrada que hay en un solo directorio, utilice lo siguiente:

```
convert-ly -e *.ly
```

Tanto los usuarios de Linux como los de MacOS X pueden usar la aplicación de terminal correspondiente, pero los usuarios de MacOS X pueden también ejecutar esta orden directamente desde el menú Compilar > Actualizar la sintaxis.

Un usuario de Windows ejecutaría la instrucción:

```
convert-ly.py -e *.ly
```

escribiéndola en un terminal de línea de órdenes o indicador del sistema que normalmente se encuentra bajo Inicio > Accessorios > Consola de órdenes o, para los usuarios de la versión 8, escribiendo en la ventana de búsqueda 'consola de órdenes'.

Para converitr todos los archivos de entrada que residen en distintos conjuntos de subdirectorios:

```
find . -name '*.ly' -exec convert-ly -e '{}' \;
```

Este ejemplo busca y convierte todos los archivos de entrada que están en el directorio actual y en todos los directorios que están dentro de él, de forma recursiva. Los archivos convertidos se colocan en el mismo directorio que sus originales renombrados. También debería funcionar para los usuarios de MacOS X, si bien solamente a través de la aplicación de terminal.

Los usuarios de Windows deben hacer lo siguiente:

```
forfiles /s /M *.ly /c "cmd /c convert-ly.py -e @file"
```

Como alternativa, se puede indicar una ruta explícita al nivel superior del directorio que contiene todos los sub-directorios que contienen archivos de entrada, mediante la opción /p:

```
forfiles /s /p C:\Documentos\MisPartituras /M *.ly /c "cmd /c convert-ly.py -e @file"
```

Si el nombre o la ruta del directorio de nivel superior contienen espacios, entonces hay que poner entre comillas la ruta completa:

```
forfiles /s /p "C:\Documentos\Mis Partituras" /M *.ly /c "cmd /c convert-ly.py -e @file
```

2.3 Opciones de la línea de órdenes para convert-ly

En general, el programa se invoca de la manera siguiente:

```
convert-ly [opción]... archivo...
```

Se pueden dar las siguientes opciones:

`-d, --diff-version-update`
: incrementar la cadena `\version` solamente si el archivo efectivamente ha cambiado. En tal caso, la cabecera de versión corresponderá a la versión siguiente al último cambio efectivo. Los números de las versiones de desarrollo se redondean hacia arriba al número de la siguiente versión estable, a no ser que fuera superior al número de la versión objetivo. Sin esa opción, la versión refleja la última conversión que se *intentó* hacer.

`-e, --edit`
: Aplicar las conversiones directamente al archivo de entrada, modificándolo in situ. El archivo original se cambia de nombre a `miarchivo.ly~`. Este archivo de copia de seguridad podría ser un archivo oculto en algunos sistemas operativos. Como alternativa, si queremos especificar un nombre distinto para el archivo actualizado sin que la tilde curva `~`, predeterminada de la opción `-e`, se añada al final del nombre del archivo antiguo, se puede en su lugar redirigir la entrada:

```
convert-ly miarchivo.ly > miarchivonuevo.ly
```

Los usuarios de Windows harán lo siguiente:

```
convert-ly.py miarchivo.ly > miarchivonuevo.ly
```

`-b, --backup-numbered`
: Cuando se usa con la opción '`-e`', numerar los archivos de copia de seguridad de forma que no se sobreescriba ninguna versión anterior. Los archivos de copia de seguridad podrían ser archivos ocultos en algunos sistemas operativos.

`-f, --from=`*versión_de_origen*
: Establece la versión desde la que convertir. Si no aparece esta opción, `convert-ly` tratará de adivinarla, basándose en el enunciado `\version` del archivo. Ejemplo: `--from=2.10.25`

`-h, --help`
: Imprimir la ayuda de utilización.

`-l` *loglevel*`, --loglevel=`*loglevel*
: Fijar el grado en que la salida es prolija a *loglevel*. Los valores posibles son `NONE` (ninguno), `ERROR` (errores), `WARNING` (advertencias), `PROGRESS` (avance; predeterminado) y `DEBUG` (depuración).

`-n, --no-version`
: Normalmente `convert-ly` añade un indicador `\version` a la salida. La especificación de esta opción lo suprime.

`-s, --show-rules`
: Mostrar todas las conversiones conocidas y salir.

`-t, --to=`*versión_final*
: Fijar explícitamente a qué `\version` convertir, en caso contrario el valor predeterminado es la versión más actual. Debe ser más alta que la versión de partida.

```
convert-ly --to=2.14.1 miarchivo.ly
```

Para actualizar fragmentos de LilyPond en archivos de texinfo, use

```
convert-ly --from=... --to=... --no-version *.itely
```

Para ver los cambios en la sintaxis de LilyPond entre dos versiones dadas, use

```
convert-ly --from=... --to=... -s
```

2.4 Problemas con convert-ly

Al ejecutar convert-ly en una ventana del Símbolo del Sistema bajo Windows sobre un archivo que tiene espacios en el nombre o en la ruta, es necesario encerrar todo el nombre del archivo de entrada con tres (!) pares de comillas:

```
convert-ly """D:/Mis partituras/Oda.ly""" > "D:/Mis partituras/nueva Oda.ly"
```

Si la orden simple `convert-ly -e *.ly` no funciona porque la instrucción expandida se hace muy larga, en vez de ello la orden `convert-ly` se puede poner dentro de un bucle. Este ejemplo para UNIX actualiza todos los documentos `.ly` del directorio actual

```
for f in *.ly; do convert-ly -e $f; done;
```

En la ventana del terminal de órdenes de Windows, la instrucción correspondiente es

```
for %x in (*.ly) do convert-ly -e """%x"""
```

No se manejan todos los cambios en el lenguaje. Sólo se puede especificar una opción de salida. La actualización automática de Scheme y los interfaces Scheme de LilyPond es bastante improbable; prepárese para trucar el código de Scheme a mano.

2.5 Conversiones manuales

En teoría, un programa como `convert-ly` debería poder tratar cualquier cambio en la sintaxis. Después de todo, un programa de ordenador interpreta las versiones antigua y nueva, por lo que otro programa de ordenador podría traducir un archivo al otro[1].

Sin embargo, el proyecto LilyPond cuenta con unos recursos limitados: no todas las conversiones se efectúan automáticamente. A continuación aparece una lista de los problemas conocidos.

```
1.6->2.0:
 No siempre convierte el bajo cifrado correctamente, específicamente cosas como {<
>}.  El comentario de Mats sobre cómo solventar el problema:
   Para poder ejecutar convert-ly
   sobre él, primero sustituí todas las apariciones de '{<' a algo mudo como '{#'
   y de forma similar sustituí '>}' con '&}'.  Después de la conversión, pude
   volver a cambiarlos de '{ #' a '{ <' y de '& }' a '> }'.
 No convierte todos los marcados de texto correctamente.  En sintaxis antigua,
 se podían agrupar varios marcados entre paréntesis, p.ej.
   -#'((bold italic) "cadena")
   Esto se convierte incorrectamente en
   -\markup{{\bold italic} "cadena"}
   en vez del correcto
   -\markup{\bold \italic "cadena"}
2.0->2.2:
 No maneja \partCombine
 No hace \addlyrics => \lyricsto, esto rompe algunas partituras con varias estrofas.
2.0->2.4:
 \magnify no se cambia por \fontsize.
    - \magnify #m => \fontsize #f, donde f = 6ln(m)/ln(2)
```

[1] Al menos, esto es posible en cualquier archivo de LilyPond que no contenga Scheme. Si hay Scheme dentro del archivo, contiene un lenguaje Turing-completo, y nos encontramos con el famoso "Problema de la parada" en informática.

```
 remove-tag no se cambia.
    - \applyMusic #(remove-tag '. . .) => \keepWithTag #'. . .
 first-page-number no se cambia.
    - first-page-number no => print-first-page-number = ##f
 Los saltos de línea en las cadenas de cabecera no se convierten.
    - \\\\  como salto de línea en las cadenas de \header  => \markup \center-align <
      "Primera línea" "Segunda línea" >
 Los terminadores de crescendo y decrescendo no se convierten.
    - \rced => \!
    - \rc => \!
2.2->2.4:
 \turnOff (usado en \set Staff.VoltaBracket = \turnOff) no se convierte
adecuadamente.
2.4.2->2.5.9
 \markup{ \center-align <{ ... }> } se tendría que convertir en:
 \markup{ \center-align {\line { ... }} }
 pero ahora, falta el \line.
2.4->2.6
 Los caracteres especiales de LaTeX como $~$ en el texto no se convierten a UTF8.
2.8
 \score{} ahora debe empezar con una expresión musical.  Cualquier otra cosa
 (en particular, \header{}) debe ir después de la música.
```

2.6 Escritura de código que contemple varias versiones

En ciertos casos, especialmente al escribir código de *bibliotecas*, es deseable dar apoyo a más de una versión de LilyPond, por encima de los cambios de sintaxis que rompen con la práctica anterior. Para hacerlo, se pueden envolver porciones de código alternativas dentro de expresiones condicionales que dependen de la versión de LilyPond que se está ejecutando actualmente. La función de Scheme `ly:version?` admite un operador de comparación *op* y una versión de referencia *ver* que se pasa como una lista de enteros con un máximo de tres elementos. Se ignoran los elementos que faltan, de forma que `'(2 20)` equivale a *cualquier* versión de la línea de 2.20. Son posibles construcciones como las siguientes:

```
#(cond
  ((ly:version? > '(2 20))
   (ly:message "Esto es código para LilyPond posterior a 2.20"))
  ((ly:version? = '(2 19 57))
   (ly:message "Esto solamente se ejecuta con LilyPond 2.19.57"))
  (else (ly:message "Esto se ejecuta en cualquier otra versión")))
```

Por lo general, esto se encontrará integrado dentro de funciones de biblioteca que permitan usar más de un tipo de sintaxis alternativas, pero también es posible usar la comparación directamente dentro de la música como en el ejemplo siguiente:

```
{
  c' d' e' f'
  #(if (ly:version? = '(2 21))
       #{ \override NoteHead.color = #red #}
       #{ \override NoteHead.color = #blue #})
  g' a' b' c''
}
```

Nota: Esta función fue introducida en LilyPond 2.19.57, por lo que no es posible hacer la comparación con versiones anteriores a esa.

3 Ejecución de `lilypond-book`

Si quiere añadir imágenes de música a un documento, puede hacerlo simplemente de la forma en que lo haría con otros tipos de imágenes. Las imágenes se crean por separado, dando como resultado una salida PostScript o imágenes PNG, y luego se incluyen en un documento de LaTeX o de HTML.

`lilypond-book` ofrece una manera de automatizar este proceso: este programa extrae los fragmentos de música del documento, ejecuta `lilypond` sobre cada uno de ellos, y devuelve como salida el documento con la música sustituida por las imágenes. Las definiciones de ancho de línea y tamaño de letra de la música se ajustan de forma que coincidan con los ajustes de su documento.

Es un programa distinto a `lilypond` propiamente dicho, y se ejecuta sobre la línea de órdenes; para ver más información, consulte Sección 1.2 [Utilización desde la línea de órdenes], página 1. Si experimenta algún problema al ejecutar `lilypond-book` sobre Windows o Mac OS X utilizando la línea de órdenes, consulte Sección “Windows” en *Información general* o Sección “MacOS X” en *Información general*.

Este procedimiento se puede aplicar a documentos de LaTeX, HTML, Texinfo o DocBook.

3.1 Un ejemplo de documento musicológico

Ciertos textos contienen ejemplos musicales. Son tratados musicales, cancioneros o manuales como este mismo. Estos textos se pueden hacer a mano, importando simplemente una imagen en formato PostScript en el editor de textos. Sin embargo, hay un procedimiento automático para reducir la carga de trabajo que esto implica los documentos de HTML, LaTeX, Texinfo y DocBook.

Un guión ejecutable llamado `lilypond-book` extrae los fragmentos de música, les da formato y vuelve a poner en su lugar la partitura resultante. A continuación presentamos un pequeño ejemplo de su utilización con LaTeX. El ejemplo contiene también texto explicativo, por lo que no vamos a comentarlo posteriormente.

Entrada

```
\documentclass[a4paper]{article}

\begin{document}

Los documentos para \verb+lilypond-book+ pueden mezclar libremente
música y texto. Por ejemplo:

\begin{lilypond}
\relative {
  c'2 e2 \tuplet 3/2 { f8 a b } a2 e4
}
\end{lilypond}

Las opciones se escriben entre corchetes.

\begin{lilypond}[fragment,quote,staffsize=26,verbatim]
  c'4 f16
\end{lilypond}

Los ejemplos grandes se pueden grabar en archivos separados e
```

```
introducirse con \verb+\lilypondfile+.

\lilypondfile[quote,noindent]{screech-and-boink.ly}

(Si es necesario, sustituya @file{screech-and-boink.ly}
por cualquier archivo @file{.ly}
situado en el mismo directorio que este archivo.)

\end{document}
```

Procesado

Guarde el código anterior como un archivo llamado `lilybook.lytex`, y luego ejecute en un terminal:

```
lilypond-book --output=out --pdf lilybook.lytex
lilypond-book (GNU LilyPond) 2.24.4
Leyendo lilybook.lytex...
...montañas de mensajes suprimidos...
Compilando lilybook.tex...
cd out
pdflatex lilybook
...montañas de mensajes suprimidos...
xpdf lilybook
(sustituya xpdf por su visor de PDF favorito)
```

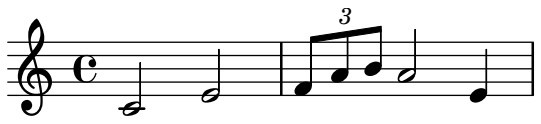
La ejecución de `lilypond-book` y `latex` crea un gran número de archivos temporales, que podrían abarrotar el directorio de trabajo. Para poner remedio a esto utilice la opción `--output=`*directorio*. Creará los archivos en un subdirectorio aparte `directorio`.

Finalmente el resultado del ejemplo de LaTeX que acabamos de mostrar[1]. Así acaba la sección del tutorial.

[1] Este tutorial se procesa con Texinfo, por lo que el ejemplo presenta un resultado en la disposición ligeramente distinto.

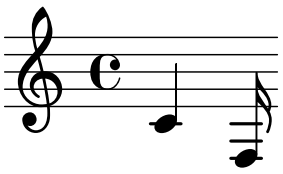
Salida

Los documentos para lilypond-book pueden mezclar libremente música y texto. Por ejemplo:

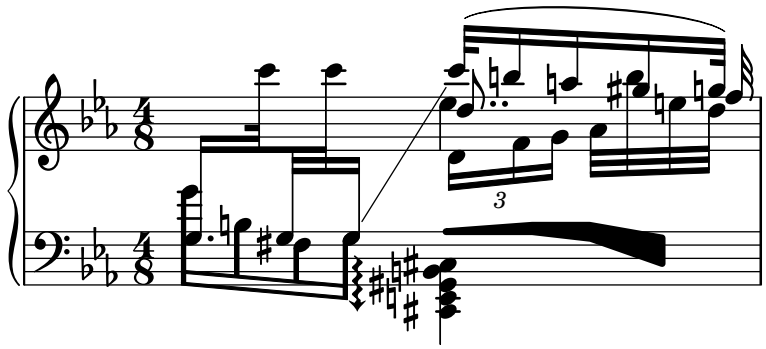


Las opciones se escriben entre corchetes.

```
c'4 f16
```



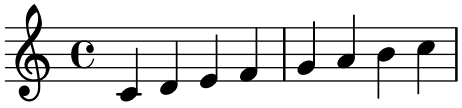
Los ejemplos grandes se pueden grabar en archivos separados e introducirse con \lilypondfile.



Si se requiere un campo tagline, ya sea predeterminado o personalizado, entonces el fragmento completo se debe incluir dentro de una construcción \book { }.

```
\book{
  \header{
    title = "Una escala en LilyPond"
  }

  \relative {
    c' d e f g a b c
  }
}
```

Una escala en LilyPond



Music engraving by LilyPond 2.24.4—www.lilypond.org

3.2 Integrar música y texto

Aquí vamos a explicar cómo integrar LilyPond con algunos otros formatos de salida.

3.2.1 LaTeX

LaTeX es el estándar de facto para la publicación en el mundo de las ciencias exactas. Está construido encima del motor de composición tipográfica TeX, proporcionando la tipografía de mejor calidad que existe.

Consulte *The Not So Short Introduction to LaTeX* (https://www.ctan.org/tex-archive/info/lshort/english/) (Introducción no tan breve a LaTeX) para ver una panorámica sobre cómo usar LaTeX.

lilypond-book aporta las instrucciones y entornos siguientes para incluir música dentro de archivos de LaTeX:

- la instrucción \lilypond{...}, donde podemos escribir directamente fragmentos cortos de código de LilyPond
- el entorno \begin{lilypond}...\end{lilypond}, donde podemos introducir directamente bloques más largos de código de LilyPond
- la instrucción \lilypondfile{...} para insertar un archivo de LilyPond
- la instrucción \musicxmlfile{...} para insertar un archivo de MusicXML, que se procesa por parte de musicxml2ly y lilypond.

En el archivo de entrada, se especifica la música con cualquiera de las instrucciones siguientes:

```
\begin{lilypond}[las,opciones,van,aquí]
  EL CÓDIGO DE LILYPOND
\end{lilypond}

\lilypond[las,opciones,van,aquí]{ EL CÓDIGO DE LILYPOND }

\lilypondfile[las,opciones,van,aquí]{archivo}

\musicxmlfile[las,opciones,van,aquí]{archivo}
```

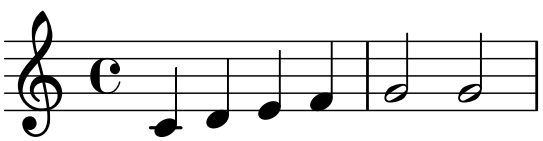
De forma adicional, \lilypondversion imprime la versión actual de lilypond.

La ejecución de lilypond-book deja como resultado un archivo que se puede procesar posteriormente con LaTeX.

A continuación mostramos algunos ejemplos. El entorno lilypond

```
\begin{lilypond}[quote,fragment,staffsize=26]
  c' d' e' f' g'2 g'2
\end{lilypond}
```

produce



La versión corta

```
\lilypond[quote,fragment,staffsize=11]{<c' e' g'>}
```

produce

Por el momento no es posible incluir llaves { o } dentro de \lilypond{}, así que esta instrucción solamente es útil con la opción fragment.

El ancho predeterminado de las líneas de música se ajusta mediante el examen de las instrucciones del preámbulo del documento, la parte del documento que está antes de \begin{document}. La instrucción lilypond-book los envía a LaTeX para averiguar la anchura del texto. El ancho de la línea para los fragmentos de música se ajusta entonces al ancho del texto. Observe que este algoritmo heurístico puede fácilmente fallar; en estos casos es necesario usar la opción line-width del fragmento de música.

Cada fragmento ejecutará los macros siguientes si han sido definidos por el usuario:

- \preLilyPondExample que se llama antes de la música,
- \postLilyPondExample que se llama después de la música,
- \betweenLilyPondSystem[1] se llama entre los sistemas si lilypond-book ha dividido el fragmento en varios archivos PostScript. Se debe definir de forma que tome un parámetro y recibirá el número de archivos ya incluidos dentro del fragmento actual. La acción predeterminada es simplemente insertar un \linebreak.

Fragmentos de código seleccionados

A veces es útil mostrar elementos de música (como ligaduras) como si continuasen más allá del final del fragmento. Esto se puede hacer dividiendo el pentagrama y suprimiendo la inclusión del resto de la salida de LilyPond.

En LaTeX, defina \betweenLilyPondSystem de tal forma que la la inclusión de otros sistemas se dé por terminada una vez que se ha alcanzado el número deseado de sistemas requeridos. Puesto que \betweenLilyPondSystem se llama en primer lugar *después* del primer sistema, incluir solamente el primer sistema es algo trivial.

```
\def\betweenLilyPondSystem#1{\endinput}

\begin[fragment]{lilypond}
  c'1\( e'( c'~ \break c' d) e f\)
\end{lilypond}
```

Si se necesita un mayor número de sistemas, se tiene que usar un condicional de TeX antes del \endinput. En este ejemplo, sustituya el '2' por el número de sistemas que quiere en la salida:

```
\def\betweenLilyPondSystem#1{
    \ifnum#1<2\else\expandafter\endinput\fi
}
```

(Dado que \endinput detiene inmediatamente el procesado del archivo de entrada actual, necesitamos \expandafter para posponer la llamada de \endinput después de ejecutar \fi de manera que la cláusula \if-\fi esté equilibrada.)

Recuerde que la definición de \betweenLilyPondSystem es efectiva hasta que TeX abandona el grupo actual (como el entorno LaTeX) o se sobreescribe por otra definición (lo que casi siempre es por el resto del documento). Para reponer la definición, escriba

```
\let\betweenLilyPondSystem\undefined
```

dentro de la fuente de LaTeX.

Se puede simplificar esto definiendo un macro de TeX:

```
\def\onlyFirstNSystems#1{
```

```
    \def\betweenLilyPondSystem##1{\ifnum##1<#1\else\endinput\fi}
}
```

y luego diciendo solamente cuántos sistemas quiere antes de cada fragmento:

```
\onlyFirstNSystems{3}
\begin{lilypond}...\end{lilypond}
\onlyFirstNSystems{1}
\begin{lilypond}...\end{lilypond}
```

Véase también

Hay opciones de línea de órdenes específicas de lilypond-book y otros detalles que conocer para procesar documentos deLaTeX véase Sección 3.4 [Invocar lilypond-book], página 35.

3.2.2 Texinfo

Texinfo es el formato estándar para la documentación del proyecto GNU. Este mismo manual es un ejemplo de documento Texinfo. Las versiones HTML, PDF e Info del manual se hacen a partir del documento Texinfo.

lilypond-book aporta las siguientes instrucciones y entornos para incluir música dentro de archivos de Texinfo:

- la instrucción @lilypond{...}, donde podemos introducir directamente fragmentos cortos de código de LilyPond
- el entorno @lilypond...@end lilypond, donde podemos escribir directamente bloques más extensos de código de LilyPond
- la instrucción @lilypondfile{...} para insertar un archivo de LilyPond
- la instrucción @musicxmlfile{...} para insertar un archivo de MusicXML, que se procesa después por parte de musicxml2ly y de lilypond.

En el archivo de entrada, la música se especifica con cualquiera de las instrucciones siguientes:

```
@lilypond[las,opciones,van,aquí]
  EL CÓDIGO DE LILYPOND
@end lilypond

@lilypond[las,opciones,van,aquí]{ EL CÓDIGO DE LILYPOND }

@lilypondfile[las,opciones,van,aquí]{archivo}

@musicxmlfile[las,opciones,van,aquí]{archivo}
```

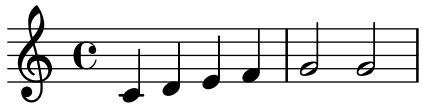
De forma adicional, @lilypondversion imprime la versión actual de lilypond.

Cuando se ejecuta lilypond-book sobre el archivo, se obtiene como resultado un archivo Texinfo (con la extensión .texi) que contiene etiquetas @image para el HTML, Info y la salida impresa. lilypond-book genera imágenes de la música en formatos EPS y PDF para usarlos en la salida impresa, y en formato PNG para usarlos en las salidas HTML e Info.

Aquí podemos ver dos ejemplos sencillos. Un entorno lilypond

```
@lilypond[fragment]
c' d' e' f' g'2 g'
@end lilypond
```

produce



La versión corta

```
@lilypond[fragment,staffsize=11]{<c' e' g'>}
```

produce

A diferencia de LaTeX, `@lilypond{...}` no genera una imagen en línea. Siempre consiste en un párrafo para ella sola.

3.2.3 HTML

lilypond-book aporta las siguientes instrucciones y entornos para incluir música dentro de archivos HTML:

- la instrucción `<lilypond ... />`, donde podemos introducir directamente fragmentos cortos de código de LilyPond
- el entorno `<lilyond>...</lilypond>`, donde podemos escribir directamente bloques más extensos de código de LilyPond
- la instrucción `<lilypondfile>...</lilypondfile>` para insertar un archivo de LilyPond
- la instrucción `<musicxmlfile>...</musicxmlfile>` para insertar un archivo de MusicXML, que se procesa después por parte de musicxml2ly y de lilypond.

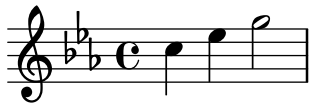
En el archivo de entrada, la música se especifica con cualquiera de las instrucciones siguientes:

```
<lilypond las opciones van aquí>
  EL CÓDIGO DE LILYPOND
</lilypond>

<lilypond las opciones van aquí: EL CÓDIGO DE LILYPOND />

<lilypondfile las opciones van aquí>archivo</lilypondfile>

<musicxmlfile las opciones van aquí>archivo</musicxmlfile>
```

Por ejemplo, podemos escribr

```
<lilypond fragment relative=2>
\key c \minor c4 es g2
</lilypond>
```

lilypond-book entonces produce un archivo HTML con las etiquetas de imagen adecuadas para los fragmentos de música:



Para imágenes en línea, utilice `<lilypond ... />`, donde las opciones están separadas de la música por el símbolo de dos puntos, por ejemplo

```
Algo de música dentro de <lilypond relative=2: a b c/> una línea
de texto.
```

Para incluir archivos externos, escriba

```
<lilypondfile opción1 opción2 ...>archivo</lilypondfile>
```

`<musicxmlfile>` usa la misma sintaxis que `<lilypondfile>`, pero sencillamente referencia un archivo de MusicXML en lugar de un archivo de LilyPond.

Para ver una lista de las opciones que utilizar con las etiquetas lilypond o lilypondfile, véase Sección 3.3 [Opciones de fragmentos de música], página 33.

De forma adicional, <lilypondversion/> imprime la versión actual de lilypond.

3.2.4 DocBook

Para insertar fragmentos de LilyPond es bueno tratar de mantener la conformidad del documento de DocBook, permitiendo así el uso de editores de DocBook, validación, etc. Así pues, no usamos etiquetas personalizadas, sólo especificamos una convención basada en los elementos estándar de DocBook.

Convenciones usuales

Para insertar toda clase de fragmentos utilizamos los elementos mediaobject y inlinemediaobject, de forma que nuestros fragmentos puedan ser formateados en línea o no en línea. Las opciones de formateo del fragmento se escriben siempre dentro de la propiedad role del elemento más interno (véanse las secciones siguientes). Las etiquetas se eligen de forma que permitan a los editores de DocBook formatear el contenido satisfactoriamente. Los archivos de DocBook que se van a procesar con lilypond-book deben tener la extensión .lyxml.

Incluir un archivo de LilyPond

Este es el caso más sencillo. Debemos usar la extensión .ly para el archivo incluido, e insertarlo como un imageobject estándar, con la estructura siguiente:

```
<mediaobject>
  <imageobject>
    <imagedata fileref="music1.ly" role="printfilename" />
  </imageobject>
</mediaobject>
```

Observe que puede usar mediaobject o inlinemediaobject como el elemento más externo, a elección suya.

Incluir código de LilyPond

Se puede incluir código de LilyPond mediante la utilización de un elemento programlisting, en que el lenguaje se establece como lilypond con la estructura siguiente:

```
<inlinemediaobject>
  <textobject>
    <programlisting language="lilypond" role="fragment verbatim staffsize=16 ragged-rig
\context Staff \with {
  \remove Time_signature_engraver
  \remove Clef_engraver}
  { c4( fis) }
    </programlisting>
  </textobject>
</inlinemediaobject>
```

Como puede ver, el elemento más externo es un mediaobject o un inlinemediaobject, y hay un elemento textobject que lleva el programlisting en su interior.

Procesar el documento de DocBook

Al ejecutar lilypond-book sobre el archivo .lyxml se creará un documento de DocBook válido que se puede procesar posteriormente con la extensión .xml. Si usa dblatex (http://dblatex.sourceforge.net), creará un archivo PDF a partir de este documento automáticamente. Para

la generación de HTML (HTML Help, JavaHelp, etc.) puede usar las hojas de estilo oficiales XSL de DocBook, aunque es posible que tenga que aplicarles algún tipo de personalización.

3.3 Opciones de fragmentos de música

Durante los próximos párrafos, una 'instrucción de LilyPond' se refiere a cualquier instrucción descrita en las secciones anteriores que se maneja por parte de `lilypond-book` para que produzca un fragmento de música. Por simplicidad, las instrucciones de LilyPond solamente se muestran en la sintaxis de LaTeX.

Observe que la cadena de opciones se analiza de izquierda a derecha; si una opción aparece varias veces, se toma la última solamente.

Están disponibles las siguientes opciones para las instrucciones de LilyPond:

`staffsize=`*altura*
: Establecer la altura del pentagrama como *altura*, medida en puntos.

`ragged-right`
: Producir líneas no justificadas por la derecha y con espaciado natural, es decir, se añade `ragged-right = ##t` al fragmento de LilyPond. Los fragmentos de una sola línea siempre se tipografían de forma predeterminada sin justificación por la derecha, a no ser que se use explícitamente la opción `noragged-right`.

`noragged-right`
: Para fragmentos de una sola línea, permitir que la longitud del pentagrama se amplíe hasta igualar la anchura de la línea, es decir, se añade `ragged-right = ##f` al fragmento de LilyPond.

`line-width`
`line-width=`*tamaño*`\`*unidades*
: Establecer el ancho de línea como *tamaño*, utilizando *unidades* como unidad. *unidades* es una de las siguientes cadenas: `cm`, `mm`, `in` o `pt`. Esta opción afecta a la salida de LilyPond (esto es, a la longitud del pentagrama del fragmento musical), no al formato del texto.

  Si se usa sin ningún argumento, se establece el ancho de la línea a un valor predeterminado (calculado con un algoritmo heurístico).

  Si no se da ninguna opción `line-width`, `lilypond-book` trata de adivinar un valor predeterminado para los entornos `lilypond` que no usan la opción `ragged-right`.

`papersize=`*cadena*
: Donde *cadena* es un tamaño del papel definido en el archivo `scm/paper.scm`, es decir, `a5`, `quarto`, `11x17`, etc.

  Los valores no definidos en el archivo `scm/paper.scm` se ignoran, se emite una advertencia y el fragmento se imprime utilizando el tamaño predeterminado `a4`.

`notime` No imprimir la indicación de compás, y desactivar las indicaciones temporales de la música (indicación del compás y líneas divisorias).

`fragment` Hacer que `lilypond-book` añada algunos códigos necesarios para que podamos escribir simplemente, por ejemplo,

```
c'4
```

  sin `\layout`, `\score`, etc.

`nofragment`
: No incluir el código adicional que completa la sintaxis de LilyPond en los fragmentos de música. Al ser la opción predeterminada, `nofragment` normalmente es redundante.

indent=*tamaño**unidades*
: Establecer el sangrado del primer sistema de pentagramas como *tamaño*, utilizando *unidades* como unidad. *unidades* es una de las siguientes cadenas: cm, mm, in o pt. Esta opción afecta a LilyPond, no al formato del texto.

noindent
: Establecer el sangrado del primer sistema de la música como cero. Esta opción afecta a LilyPond, no al formato del texto. Puesto que el valor predeterminado es que no haya ningún sangrado, noindent normalmente es redundante.

quote
: Reducir la longitud de la línea de un fragmento musical en $2 * 0.4$ in (pulgadas) y colocar la salida dentro de un bloque de cita (quotation). El valor de '0.4 in' se puede controlar con la opción exampleindent.

exampleindent
: Establecer la longitud del sangrado que la opción quote aplica al fragmento musical.

relative
relative=*n*
: Usar el modo de octava relativa. De forma predeterminada, las notas se especifican con relación al Do central. El argumento entero opcional especifica la octava de la nota inicial, donde el valor predeterminado 1 es el Do central. La opción relative sólo funciona cuando está establecida la opción fragment, de manera que fragment viene implicada automáticamente por relative, independientemente de la presencia de fragment o de nofragment en la fuente.

LilyPond utiliza también lilypond-book para producir su propia documentación. Para hacerlo, están a nuestra disposición ciertas opciones algo esotéricas para los fragmentos musicales.

verbatim
: El argumento de una instrucción de LilyPond se copia al archivo de salida y se incluye dentro de un bloque «verbatim» o preformateado, seguido del texto que se escriba con la opción intertext (que no funciona aún); después se imprime la música en sí. Esta opción no funciona bien con \lilypond{} si forma parte de un párrafo.

  Si se usa la opción verbatim dentro de una instrucción lilypondfile, es posible incluir con estilo preformateado sólo una parte del archivo fuente. Si el archivo de código fuente contiene un comentario que contiene 'begin verbatim' (sin las comillas), la cita del bloque de estilo preformateado empezará después de la última vez que aparezca este comentario; de forma similar, la cita del bloque preformateado se detendrá justo antes de la primera vez que aparezca un comentario que contenga 'end verbatim', si lo hay. En el siguiente ejemplo de código fuente, la música se interpreta en el modo relativo, pero la cita preformateada no presentará el bloque relative, es decir

```
\relative { % begin verbatim
  c'4 e2 g4
  f2 e % end verbatim
}
```

  se imprime como un bloque preformateado como

```
  c4 e2 g4
  f2 e
```

  Si queremos traducir los comentarios y los nombres de variable en la salida literal pero no en el código fuente, podemos establecer el valor de la variable de entorno LYDOC_LOCALEDIR a la ruta de un directorio; este directorio debe contener un árbol de catálogos de mensajes .mo con lilypond-doc como dominio.

texidoc (Sólo para la salida de Texinfo.) Si se llama a lilypond con la opción --header=texidoc, y el archivo que se procesa se llama fulanito.ly, crea un archivo fulanito.texidoc si existe un campo texidoc dentro del bloque \header de cabecera. La opción texidoc hace que lilypond-book incluya estos archivos, añadiendo su contenido como un bloque de documentación inmediatamente antes del fragmento musical (pero fuera del entorno example generado por la opción quote).

Suponiendo que el archivo fulanito.ly contiene

```
\header {
  texidoc = "Este archivo es un ejemplo de una sola nota."
}
{ c'4 }
```

y que tenemos lo siguiente en nuestro documento de Texinfo prueba.texinfo

```
@lilypondfile[texidoc]{fulanito.ly}
```

la siguiente orden da como salida el resultado esperado:

```
lilypond-book --pdf --process="lilypond \
  --header=texidoc" test.texinfo
```

La mayoría de los documentos de prueba de LilyPond (en el directorio input de la distribución) son pequeños archivos .ly que tienen exactamente este aspecto.

Por motivos de localización de idioma, si el documento de Texinfo contiene @documentlanguage *LANG* y la cabecera de loquesea.ly contiene un campo texidoc*LANG*, y lilypond se ejecuta con --header=texidoc*LANG*, entonces se incluirá loquesea.texidoc*LANG* en lugar de loquesea.texidoc.

doctitle (Sólo para la salida de Texinfo.) Esta opción funciona de forma parecida a la opción texidoc: si lilypond se llama con la opción --header=doctitle, y el archivo que procesar se llama loquesea.ly y contiene un campo doctitle en el bloque \header, crea un archivo loquesea.doctitle. Cuando se usa la opción doctitle, el contenido de loquesea.doctitle, que debería ser una línea única de *texto*, se inserta en el documento de Texinfo como @lydoctitle *texto*. @lydoctitle debe ser un macro definido en el documento de Texinfo. La misma indicación referida al procesado de texidoc con idiomas localizados se aplica a doctitle.

nogettext (Sólo para la salida de Texinfo.) No traducir los comentarios y nombres de variable en el fragmento de código literal citado.

printfilename Si un archivo de entrada de LilyPond se incluye con \lilypondfile, imprimir el nombre del archivo inmediatamente antes del fragmento musical. Para la salida HTML, esto es un enlace. Sólo se imprime el nombre base del archivo, es decir, se elimina la parte del directorio de la ruta del archivo.

3.4 Invocar lilypond-book

lilypond-book produce un archivo con una de las siguientes extensiones: .tex, .texi, .html o .xml, dependiendo del formato de salida. Todos los archivos .tex, .texi y .xml necesitan un procesado posterior.

Instrucciones específicas de formato

LaTeX

Hay dos formas de procesar el documento en LaTeX para su impresión o publicación: hacer un archivo PDF directamente con PDFLaTeX, o generar un archivo PostScript con LaTeX a través

de un traductor de DVI a PostScript como dvips. la primera forma es más sencilla y es la que se recomienda[1], y cualquiera que sea el método que utilice, podrá convertir fácilmente entre PostScript y PDF con herramientas como ps2pdf y pdf2ps que vienen incluidas con Ghostscript.

Para producir un archivo PDF por medio de PDFLaTeX, utilice:

```
lilypond-book --pdf miarchivo.pdftex
pdflatex miarchivo.tex
```

Para producir una salida PDF por medio de LaTeX/dvips/ps2pdf:

```
lilypond-book miarchivo.lytex
latex miarchivo.tex
dvips -Ppdf miarchivo.dvi
ps2pdf miarchivo.ps
```

El archivo .dvi creado por este proceso no contiene las cabezas de las notas. Esto es normal; si sigue las instrucciones, las cabezas aparecerán en los archivos .ps y .pdf.

La ejecución de dvips puede dar como resultado algunas advertencias sobre las fuentes tipográficas; son inocuas y se pueden ignorar. Si está ejecutando latex en modo de dos columnas, recuerde añadir -t landscape a las opciones de dvips.

Entornos tales como:

```
\begin{lilypond} ... \end{lilypond}
```

no se interpretan por parte de LaTeX. En su lugar, el programa lilypond-book extrae estos 'entornos' como archivos independientes y ejecuta LilyPond sobre ellos. Después, toma las imágenes resultantes y crea un archivo .tex en el que los macros \begin{lilypond}...\end{lilypond} se sustituyen por instrucciones de 'inserción de gráficos'. A continuación, se ejecuta LaTeX (aunque LaTeX se ha ejecutado anteriormente, lo habrá sido sobre un archivo 'vacío' para calcular cosas como el \linewidth).

Advertencias y problemas conocidos

La instrucción \pageBreak no funciona dentro de un entorno \begin{lilypond} ... \end{lilypond}.

Muchas variables del bloque \paper tampoco funcionan dentro de un entorno \begin{lilypond} ... \end{lilypond}. Use \newcommand con \betweenLilyPondSystem en el preámbulo:

```
\newcommand{\betweenLilyPondSystem}[1]{\vspace{36mm}\linebreak}
```

Texinfo

Para producir un documento de Texinfo (en cualquier formato de salida), siga el procedimiento normal para Texinfo, esto es: o bien llame a texi2pdf o a texi2dvi o a makeinfo, según el formato de la salida que quiera crear. Consulte la documentación de Texinfo para ver más detalles.

Opciones de la línea de órdenes

lilypond-book acepta las siguientes opciones de la línea de órdenes:

`-f` *formato*
`--format=`*formato*

Especificar el tipo del documento que se va a procesar: html, latex, texi (predeterminado) o docbook. Si falta esta opción, lilypond-book tratará de detectar el formato automáticamente, véase Sección 3.5 [Extensiones de nombres de archivo], página 38. Por el momento, texi es lo mismo que texi-html.

[1] Observe que PDFLaTeX y LaTeX podrían no ser utilizables para compilar cualquier documento LaTeX, y es por lo que explicamos las dos formas.

`-F` *filtro*
`--filter=`*filtro*
: Conducir los fragmentos a través de *filter* por medio de una tubería. `lilypond-book` no obedecerá –filter y –process al mismo tiempo. Por ejemplo,

```
lilypond-book --filter='convert-ly --from=2.0.0 -' mi-libro.tely
```

`-h`
`--help`
: Imprimir un breve mensaje de ayuda.

`-I` *directorio*
`--include=`*directorio*
: Añadir *directorio* a la ruta de inclusión. `lilypond-book` busca también los fragmentos ya compilados en la ruta de inclusión, y no los vuelve a escribir en el directorio de salida, así que en ciertos casos es necesario invocar instrucciones de procesado posteriores como `makeinfo` o `latex` con las mismas opciones `-I` *directorio*.

`-l` *nivel_de_registro*
`--loglevel=`*nivel_de_registro*
: Fijar el nivel en que la salida es prolija, al valor *nivel_de_registro*. Los valores posibles son `NONE` (nada), `ERROR` (errores), `WARNING` (advertencias), `PROGRESS` (avance; predeterminado) y `DEBUG` (depuración). Si esta opción no se utiliza, y la variable de entorno `LILYPOND_BOOK_LOGLEVEL` está establecida, se usa su valor como el nivel de registro.

`-o` *directorio*
`--output=`*directorio*
: Colocar los archivos generados en el *directorio*. La ejecución de `lilypond-book` genera montañas de pequeños archivos que luego procesará LilyPond. Para evitar toda esta parafernalia en el mismo directorio que la fuente, utilice la opción `--output`, y cambie a este directorio antes de ejecutar `latex` o `makeinfo`.

```
lilypond-book --output=out miarchivo.lytex
cd out
...
```

`--skip-lily-check`
: Evitar el fracaso si no se encuentra ninguna salida de lilypond. Se usa para la documentación de LilyPond en formato Info sin imágenes.

`--skip-png-check`
: Evitar el fracaso si no se encuentran las imágenes PNG de los archivos EPS. Se usa para la documentación de LilyPond en formato Info sin imágenes.

`--lily-output-dir=`*directorio*
: Escribir archivos lily-XXX en el directorio *directorio*, enlazar en el directorio de `--output`. Use esta opción para ahorrar tiempo de construcción para documentos de distintos directorios que comparten muchos fragmentos idénticos de código.

`--lily-loglevel=`*nivel_de_registro*
: Fijar el nivel en que la salida es prolija para las llamadas de la instrucción invocada `lilypond`, al valor *nivel_de_registro*. Los valores posibles son `NONE` (nada), `ERROR` (errores), `WARNING` (advertencias), `BASIC_PROGRESS` (avance básico), `PROGRESS` (avance), `INFO` (información; predeterminado) y `DEBUG` (depuración). Si no se utiliza esta opción y la variable de entorno `LILYPOND_LOGLEVEL` está establecida, su valor se usa como nivel de registro.

`--info-images-dir=`*directorio*
: Dar formato a la salida de Texinfo de manera que Info busque las imágenes de música en *directorio*.

`--latex-program=`*prog*
: Ejecutar el programa `prog` en vez de `latex`. Esto es útil si nuestro documento se procesa con `xelatex`, por ejemplo.

`--left-padding=`*cantidad*
: Rellenar las cajas EPS en esta medida, alrededor. *cantidad* se mide en milímetros, y es 3.0 como valor predeterminado. Esta opción se debe usar si las líneas de música están muy pegadas al margen derecho.

  El ancho de un sistema que está muy ajustado dentro de su rectángulo puede variar, debido a los elementos de notación que están pegados al margen izquierdo, como los números de compás y el nombre del instrumento. Esta opción acorta todas las líneas y las mueve a la derecha en la misma medida.

`-P` *instrucción*
`--process=`*instrucción*
: Procesar los fragmentos de LilyPond utilizando *instrucción*. La instrucción predeterminada es `lilypond`. `lilypond-book` no obedecerá a `--filter` y a `--process` al mismo tiempo.

`--pdf` Crear archivos PDF para su uso con PDFLaTeX.

`--redirect-lilypond-output`
: De forma predeterminada, la salida se imprime por la consola. Esta opción redirecciona toda la salida hacia archivos de registro situados en el mismo directorio que los archivos fuente.

`--use-source-file-names`
: Escribir los archivos de salida de los fragmentos de música con el mismo nombre de base que su archivo fuente. Esta opción sólo funciona para fragmentos incluidos con `lilypondfile` y sólo si los directorios determinados por las opciones `--output-dir` y `--lily-output-dir` son distintos.

`-V`
`--verbose`
: Ser prolijo. Equivale a `--loglevel=DEBUG`.

`-v`
`--version`
: Imprimir la información de la versión.

Advertencias y problemas conocidos

La instrucción de Texinfo `@pagesizes` no se interpreta. De forma similar, las instrucciones de LaTeX que cambian los márgenes y anchos de línea después del preámbulo se ignoran.

Sólo se procesa el primer `\score` de un bloque LilyPond.

3.5 Extensiones de nombres de archivo

Puede usar cualquier extensión para el nombre del archivo de entrada, pero si no usa la extensión recomendada para un formato en particular tendrá que especificar manualmente el formato de salida; para ver más detalles, consulte Sección 3.4 [Invocar lilypond-book], página 35. En caso contrario, `lilypond-book` selecciona automáticamente el formato de salida basándose en la extensión del nombre del archivo de entrada.

extensión	formato de salida
.html	HTML
.htmly	HTML
.itely	Texinfo
.latex	LaTeX
.lytex	LaTeX
.lyxml	DocBook
.tely	Texinfo
.tex	LaTeX
.texi	Texinfo
.texinfo	Texinfo
.xml	HTML

Si usa la misma extensión para el archivo de entrada que la que usa lilypond-book para el archivo de salida, y si el archivo de entrada está en el mismo directorio que el directorio de trabajo de lilypond-book, debe usar la opción --output para que funcione lilypond-book, pues en caso contrario saldrá con un mensaje de error como "La salida sobreescribirá al archivo de entrada".

3.6 Plantillas de lilypond-book

Estas plantillas se usan para lilypond-book. Si no está familiarizado con este programa, consulte Sección "lilypond-book" en *Utilización del Programa*.

3.6.1 LaTeX

Podemos insertar fragmentos de LilyPond dentro de un documento de LaTeX.

```
\documentclass[]{article}

\begin{document}

Texto normal en LaTeX.

\begin{lilypond}
\relative {
  a'4 b c d
}
\end{lilypond}

Más texto en LaTeX, y las opciones dentro de los corchetes.

\begin{lilypond}[fragment,relative=2,quote,staffsize=26,verbatim]
d4 c b a
\end{lilypond}
\end{document}
```

3.6.2 Texinfo

Podemos insertar fragmentos de LilyPond dentro de Texinfo; de hecho, todo el presente manual está escrito en Texinfo.

```
\input texinfo @node Top
@top

Texto en Texinfo
```

```
@lilypond
\relative {
  a4 b c d
}
@end lilypond

Más texto en Texinfo, y las opciones dentro de los corchetes.

@lilypond[verbatim,fragment,ragged-right]
d4 c b a
@end lilypond

@bye
```

3.6.3 html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<body>

<p>
Los documentos para lilypond-book pueden mezclar música y texto libremente.  Por
ejemplo,
<lilypond>
\relative {
  a'4 b c d
}
</lilypond>
</p>

<p>
Otro poco de lilypond, esta vez con opciones:

<lilypond fragment quote staffsize=26 verbatim>
a4 b c d
</lilypond>
</p>

</body>
</html>
```

3.6.4 xelatex

```
\documentclass{article}
\usepackage{ifxetex}
\ifxetex
%xetex specific stuff
\usepackage{xunicode,fontspec,xltxtra}
\setmainfont[Numbers=OldStyle]{Times New Roman}
\setsansfont{Arial}
```

```
\else
%Esto se puede dejar vacío si no vamos a utilizar pdftex
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage{mathptmx}%Times
\usepackage{helvet}%Helvetica
\fi
%Aquí insertamos todos los paquetes que pdftex también entiende
\usepackage[ngerman,finnish,english]{babel}
\usepackage{graphicx}

\begin{document}
\title{Un documento breve con LilyPond y xelatex}
\maketitle

Las instrucciones \textbf{font} normales dentro del \emph{texto}
funcionan, porque \textsf{están contempladas por \LaTeX{} y
XeteX.}  Si queremos usar instrucciones específicas como
\verb+\XeTeX+, debemos incluirlas de nuevo dentro de un entorno
\verb+\ifxetex+.  Podemos utilizar esto para imprimir la
instrucción \ifxetex \XeTeX{} \else XeTeX \fi que no es conocida
para el \LaTeX\ normal.

Dentro del texto normal podemos utilizar instrucciones de LilyPond
fácilmente, de esta forma:

\begin{lilypond}
{a2 b c'8 c' c' c'}
\end{lilypond}

\noindent
y así sucesivamente.

La fuente tipográfica de los fragmentos, establecida con LilyPond,
tendrá que establecerse desde dentro del fragmento. Para esto
puede leer la parte de lilypond-book en el manual de utilización.

\selectlanguage{ngerman}
Auch Umlaute funktionieren ohne die \LaTeX -Befehle, wie auch alle
anderen
seltsamen Zeichen: __ ______, wenn sie von der Schriftart
unterst__tzt werden.
\end{document}
```

3.7 Compartir el índice general

Estas funciones ya existen en el paquete OrchestralLily:

```
https://repo.or.cz/w/orchestrallily.git
```

Para conseguir más flexibilidad en el manejo del texto, algunos usuarios prefieren exportar la el índice general o tabla de contenidos desde lilypond y leerla dentro de LaTeX.

Exportación del índice general desde LilyPond

Esto supone que nuestra partitura tiene varios movimientos dentro del mismo archivo de salida de lilypond.

```
#(define (oly:create-toc-file layout pages)
  (let* ((label-table (ly:output-def-lookup layout 'label-page-table)))
    (if (not (null? label-table))
      (let* ((format-line (lambda (toc-item)
             (let* ((label (car toc-item))
                    (text  (caddr toc-item))
                    (label-page (and (list? label-table)
                                     (assoc label label-table)))
                    (page (and label-page (cdr label-page))))
               (format #f "~a, section, 1, {~a}, ~a" page text label))))
             (formatted-toc-items (map format-line (toc-items)))
             (whole-string (string-join formatted-toc-items ",\n"))
             (output-name (ly:parser-output-name))
             (outfilename (format #f "~a.toc" output-name))
             (outfile (open-output-file outfilename)))
        (if (output-port? outfile)
            (display whole-string outfile)
            (ly:warning (G_ "Unable to open output file ~a for the TOC information") outfilename))
        (close-output-port outfile)))))

\paper {
  #(define (page-post-process layout pages) (oly:create-toc-file layout pages))
}
```

Importación del índice general dentro de LaTeX

En LaTeX, la cabecera debe incluir lo siguiente:

```
\usepackage{pdfpages}
\includescore{nombredelapartitura}
```

donde \includescore está definido como:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% \includescore{PossibleExtension}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Leer las entradas del índice general para un archivo PDF
% a partir del archivo .toc correspondeiente.
% Esto requiere bastantes trucos de latex, porque leer cosas de un archivo
% e insertarlo dentro de los argumentos de un macro no es posible
% fácilmente.

% Solución de Patrick Fimml en el canal #latex el 18 de abril de 2009:
% \readfile{filename}{\variable}
% lee el contenido del archivo en \variable (no definida si el
% archivo no existe)
\newread\readfile@f
\def\readfile@line#1{%
{\catcode`\^^M=10\global\read\readfile@f to \readfile@tmp}%
\edef\do{\noexpand\g@addto@macro{\noexpand#1}{\readfile@tmp}}\do%
\ifeof\readfile@f\else%
\readfile@line{#1}%
\fi%
}
\def\readfile#1#2{%
\openin\readfile@f=#1 %
\ifeof\readfile@f%
\typeout{No TOC file #1 available!}%
\else%
\gdef#2{}%
\readfile@line{#2}%
\fi
```

```
\closein\readfile@f%
}%


\newcommand{\includescore}[1]{
\def\oly@fname{\oly@basename\@ifmtarg{#1}{}{_#1}}
\let\oly@addtotoc\undefined
\readfile{\oly@xxxxxxxxx}{\oly@addtotoc}
\ifx\oly@addtotoc\undefined
\includepdf[pages=-]{\oly@fname}
\else
\edef\includeit{\noexpand\includepdf[pages=-,addtotoc={\oly@addtotoc}]
{\oly@fname}}\includeit
\fi
}
```

3.8 Métodos alternativos para mezclar texto y música

Otras formas de mezclar texto y música (sin lilypond-book) se estudian en Sección 4.4.3 [Otros programas], página 52.

4 Programas externos

LilyPond es capaz de interactuar con otros programas de diversas maneras.

4.1 Apuntar y pulsar

«Point and click» (apuntar y pulsar con el ratón) le da la posibilidad de localizar notas del código de entrada pulsando sobre ellas dentro del visor de PDF. Ello facilita encontrar los fragmentos de la entrada que producen algún tipo de error en la partitura.

4.1.1 Configuración del sistema

Cuando esta funcionalidad está activada, LilyPond añade enlaces a los archivos PDF y SVG. Estos enlaces se envían al navegador de web, que a su vez abre un editor de texto con el cursor situado en el lugar correcto.

Para conseguir que esta cadena funcione, tiene que configurar el visor de PDF de forma que siga los enlaces usando el guión `lilypond-invoke-editor` proporcionado con LilyPond.

El programa `lilypond-invoke-editor` es un pequeño programa de apoyo. Invoca un editor para las URIs especiales de `textedit`, y lanza un navegador de web para el resto. Examina las variables de entorno `EDITOR` y `LYEDITOR` para determinar y lanzar el editor favorito seleccionado. `LYEDITOR` tiene prioridad sobre `EDITOR`, por lo que recomendamos usar el primero especialmente si va a utilizar un editor en la consola y otro editor para la funcionalidad Apuntar y Pulsar de LilyPond.

Cada uno de los editores puede puede tener una sintaxis distinta para abrir un archivo en una línea y columna específicas. Para la conveniencia del usuario, LilyPond incorpora instrucciones ya preparadas para varios editores, relacionados en `scm/editor.scm`. Esto significa que puede limitarse a escribir el nombre del binario del editor, p. ej.:

```
export LYEDITOR=atom
```

lo que invocará a:

```
atom %(file)s:%(line)s:%(column)s
```

donde `%(file)s`, `%(line)s` y `%(column)s` se sustituyen por el archivo, la línea y la columna respectivamente.

Para usar un editor que no está relacionado en `scm/editor.scm`, debe encontrar su sintaxis específica y asignar la instrucción al completo que corresponde a `LYEDITOR`. He aquí un ejemplo para el editor Visual Studio Code:

```
export LYEDITOR="code --goto %(file)s:%(line)s:%(column)s"
```

> **Nota:** Si elige Emacs, se necesita una configuración adicional. Debe añadir la línea `(server-start)` a su archivo `~/.emacs`, pues en caso contrario cada pulsación sobre un objeto del PDF abrirá una nueva ventana de Emacs.

Uso de Xpdf

Para Xpdf sobre Unix, lo siguiente debe estar presente en `xpdfrc`. En Unix, este archivo se encuentra o bien en `/etc/xpdfrc` o como `$HOME/.xpdfrc`.

```
urlCommand      "lilypond-invoke-editor %s"
```

Si está usando Ubuntu, probablemente la versión de Xpdf instalada en su sistema efectúe paradas abruptas en cada documento PDF: este estado se viene prolongando desde hace varios años y se debe a la falta de correspondencia entre bibliotecas. Lo mejor que puede hacer en

vez de ello es instalar un paquete ‘xpdf’ actualizado y el paquete ‘libpoppler’ correspondiente procedente de Debian. Una vez haya comprobado que funciona, puede usar

```
sudo apt-mark hold xpdf
```

con el objeto de evitar que Ubuntu lo sobreescriba con la siguiente ‘actualización’ de su paquete defectuoso.

Uso de GNOME 2

Para usar GNOME 2 (y los visores de documentos PDF integrados en él), el conjuro para informar al sistema acerca de la URI ‘textedit:’ es:

```
gconftool-2 -t string -s /desktop/gnome/url-handlers/textedit/command "lilypond-invoke-editor %s"
gconftool-2 -s /desktop/gnome/url-handlers/textedit/needs_terminal false -t bool
gconftool-2 -t bool -s /desktop/gnome/url-handlers/textedit/enabled true
```

Después de esta invocación,

```
gnome-open textedit:///etc/issue:1:0:0
```

debería llamar a lilypond-invoke-editor para abrir archivos.

Uso de GNOME 3

En GNOME 3, las URIs se manejan por parte de la capa ‘gvfs’ en vez de por ‘gconf’. Debe crear un archivo en un directorio local como /tmp con el nombre lilypond-invoke-editor.desktop y que tenga el siguiente contenido:

```
[Desktop Entry]
Version=1.0
Name=lilypond-invoke-editor
GenericName=Textedit URI handler
Comment=URI handler for textedit:
Exec=lilypond-invoke-editor %u
Terminal=false
Type=Application
MimeType=x-scheme-handler/textedit;
Categories=Editor
NoDisplay=true
```

y luego ejecute las instrucciones

```
xdg-desktop-menu install ./lilypond-invoke-editor.desktop
xdg-mime default lilypond-invoke-editor.desktop x-scheme-handler/textedit
```

Tras esta invocación,

```
gnome-open textedit:///etc/issue:1:0:0
```

debería llamar a lilypond-invoke-editor para abrir archivos.

Configuración adicional para Evince

Si gnome-open funciona, pero Evince aún rehúsa abrir enlaces de Apuntar y pulsar a causa de permisos denegados, quizá tenga que cambiar el perfil de Apparmor de Evince, que controla el tipo de acciones que se le permite realizar a Evince.

Para Ubuntu, el proceso es editar el archivo /etc/apparmor.d/local/usr.bin.evince y añadir al final las siguientes líneas:

```
# Para enlaces de Textedit
/usr/local/bin/lilypond-invoke-editor Cx -> sanitized_helper,
```

Después de añadir estas líneas, ejecute

```
sudo apparmor_parser -r -T -W /etc/apparmor.d/usr.bin.evince
```

Ahora Evince debería poder abrir enlaces de Apuntar y pulsar. Para otros visores, es probable que funcionen configuraciones similares.

Habilitar la opción de apuntar y pulsar

En LilyPond, la funcionalidad de Apuntar y pulsar está habilitada de forma predeterminada cuando se crean archivos de salida en formato PDF o SVG.

Los enlaces de apuntar y pulsar aumentan significativamente el tamaño de los archivos de salida. Para reducir el tamaño de los archivos PDF (y PS), la posibilidad de apuntar y pulsar se puede desactivar escribiendo

```
\pointAndClickOff
```

dentro de un archivo `.ly`. Se puede activar explícitamente la posibilidad de apuntar y pulsar con

```
\pointAndClickOn
```

De forma alternativa, puede desactivar la posibilidad de apuntar y pulsar con una opción de la línea de órdenes:

```
lilypond -dno-point-and-click archivo.ly
```

> **Nota:** Recuerde desactivar siempre la posibilidad Apuntar y pulsar en cualquier archivo de LilyPond que vaya a ser distribuido, para evitar incluir información de rutas de archivo relativas a su equipo dentro del archivo PDF, lo que puede plantear un problema de seguridad.

Apuntar y pulsar selectivo

Para ciertas aplicaciones interactivas podría ser deseable incluir solamente ciertos elementos dentro de la función de apuntar y pulsar. Por ejemplo, si alguien quisiera crear una aplicación que reprodujese audio o vídeo empezando por una nota concreta, sería incómodo que la pulsación sobre la nota produjese las coordenadas de apuntar y pulsar de una alteración accidental o de una ligadura de expresión que estuviese sobre dicha nota.

Esto se puede controlar indicando qué eventos incluir:

- Codificado de forma fija dentro del archivo `.ly`:

```
\pointAndClickTypes #'note-event
\relative {
  c'2\f( f)
}
```

  o bien

```
#(ly:set-option 'point-and-click 'note-event)
\relative {
  c'2\f( f)
}
```

- Línea de órdenes:

```
lilypond -dpoint-and-click=note-event   example.ly
```

Se pueden incluir varios eventos:

- Codificado de forma fija dentro del archivo `.ly`:

```
\pointAndClickTypes #'(note-event dynamic-event)
\relative {
  c'2\f( f)
}
```

o bien

```
#(ly:set-option 'point-and-click '(note-event dynamic-event))
\relative {
  c'2\f( f)
}
```

- Línea de órdenes:

```
lilypond \
  -e"(ly:set-option 'point-and-click '(note-event dynamic-event))" \
  example.ly
```

4.2 Apoyo respecto de los editores de texto

Existe apoyo por parte de varios editores de texto para LilyPond.

Modo de Emacs

Emacs tiene un `lilypond-mode`, que proporciona autocompleción de teclado, sangrado, compensación de paréntesis específica de LilyPond y resaltado de sintaxis con colores, útiles combinaciones de teclas para compilar y leer los manuales de LilyPond utilizando Info. Si el `lilypond-mode` no está instalado en su sistema, siga leyendo.

Está incluido un modo de Emacs para escribir música y ejecutar LilyPond, en el archivo del código fuente dentro del directorio `elisp`. Haga `make install` para instalarlo dentro de *elispdir*. El archivo `lilypond-init.el` se debe situar en *load-path*`/site-start.d/` o añadirse a su `~/.emacs` o `~/.emacs.el`.

Como usuario, puede querer añadir su ruta a las fuentes (p.ej. `~/site-lisp/`) a su *load-path* añadiendo la siguiente línea (modificada) a su `~/.emacs`

```
(setq load-path (append (list (expand-file-name "~/site-lisp")) load-path))
```

Modo de Vim

Para Vim (`https://www.vim.org`), se proporcionan para su uso con LilyPond un plug-in o complemento para el tipo de archivo, un modo de sangrado y un modo de resaltado de sintaxis. Para habilitar todas estas posibilidades, cree (o modifique) su archivo `$HOME/.vimrc` de manera que contenga estas tres líneas en el mismo orden:

```
filetype off
set runtimepath+=/usr/local/share/lilypond/current/vim/
filetype on
syntax on
```

Si LilyPond no está instalado en el directorio `/usr/local/`, cambie esta ruta de una forma adecuada. Este asunto se trata en Sección “Otras fuentes de información” en *Manual de Aprendizaje*.

Otros editores

Otros editores (de texto así como gráficos) tienen apoyo para LilyPond, pero sus archivos de configuración especiales no se distribuyen con LilyPond. Debe consultar la documentación de estos programas para obtener más información. Estos editores se encuentran relacionaddos en Sección “Entornos mejorados” en *Información general*.

4.3 Conversión desde otros formatos

También se puede escribir la música a través de su importación desde otros formatos. Este capítulo trata de documentar las herramientas incluidas en la distribución que permiten hacerlo. Existen otras herramientas que producen código de entrada de LilyPond, como por ejemplo

secuenciadores con interfaz gráfico y convertidores de XML. Consulte el website (`https://lilypond.org`) para ver más detalles.

Son programas distintos a `lilypond` propiamente dicho, y se ejecutan desde la línea de órdenes; consulte Sección 1.2 [Utilización desde la línea de órdenes], página 1, para ver más información. Si tiene MacOS 10.3 o 10.4 y tiene problemas para ejecutar alguno de estos guiones, p.ej. `convert-ly`, consulte Sección "MacOS X" en *Información general*.

Advertencias y problemas conocidos

Por desgracia no disponemos de los recursos necesarios para poder mantener estos programas; le rogamos que los tome "tal cual están". Se agradecerá el envío de parches correctores, pero los informes de fallo casi con certeza no se resolverán a medio plazo.

4.3.1 Invocar `midi2ly`

`midi2ly` traduce un archivo MIDI de tipo 1 a un archivo de código fuente de LilyPond.

El MIDI (Music Instrument Digital Interface, Interfase Digital para Instrumentos Musicales) es un estándar para instrumentos digitales: especifica la interconexión física, un protocolo en serie y un formato de archivo. El formato de archivo MIDI es un formato estándar de facto para exportar música de otros programas, por lo que esta posibilidad puede ser de utilidad al importar archivos de un programa que tiene un convertidor para un formato directo.

`midi2ly` convierte las pistas en contextos de Sección "Staff" en *Referencia de Funcionamiento Interno*) y los canales en contextos de Sección "Voice" en *Referencia de Funcionamiento Interno*. Se utiliza el modo relativo para las alturas, y las duraciones se escriben solamente cuando es necesario.

Es posible grabar un archivo MIDI usando un teclado digital y convertirlo después a `.ly`. Sin embargo, los intérpretes humanos no son lo suficientemente exactos desde el punto de vista rítmico como para hacer que sea trivial la conversión de MIDI a LY. Si se invoca con las opciones de cuantización (`-s` y `-d`), `midi2ly` intenta compensar los errores de medida, pero no lo hace muy bien. Por ello, no se recomienda usar `midi2ly` para archivos midi generados por seres humanos.

Se invoca a partir de la línea de órdenes como sigue:

```
midi2ly [opción]... archivo_midi
```

Observe que al decir 'línea de órdenes', queremos decir la línea de órdenes del sistema operativo. Consulte Sección 4.3 [Conversión desde otros formatos], página 47, para ver más información sobre esto.

`midi2ly` contempla las siguientes opciones:

`-a, --absolute-pitches`
: Imprimir alturas absolutas.

`-d, --duration-quant=`*DURACIÓN*
: Cuantizar las duraciones a *DURACIÓN*.

`-e, --explicit-durations`
: Imprimir valores de figura explícitos.

`-h, --help`
: Mostrar un resumen de las instrucciones de utilización.

`-k, --key=`*alteración*`[:`*minor*`]`
: Establecer la tonalidad predeterminada. *alteración* > 0 establece el número de sostenidos; *alteración* < 0 establece el número de bemoles. Se indica una tonalidad menor mediante `:1`.

`-o, --output=`*archivo*
: Escribir la salida en *archivo*.

`-s, --start-quant=`*DURACIÓN*
: Cuantizar el inicio de las notas a *DURACIÓN*.

`-t, --allow-tuplet=`*DURACIÓN*`*`*NUMERADOR*`/`*DENOMINADOR*
: Permitir duraciones de grupos especiales *DURACIÓN***NUMERADOR*/*DENOMINADOR*.

`-v, --verbose`
: Ser prolijo en comentarios.

`-V, --version`
: Imprimir el número de la versión.

`-w, --warranty`
: Presentar la garantía y el copyright.

`-x, --text-lyrics`
: Tratar todos los textos como letra de la canción.

Advertencias y problemas conocidos

Las notas superpuestas en un arpegio no se procesarán correctamente. La primera nota se lee y el resto se ignoran. Aplique la misma duración a todas las notas y añada marcas de fraseo o indicaciones de pedal.

4.3.2 Invocar `musicxml2ly`

MusicXML (`http://www.musicxml.org/`) es un dialecto del XML para representar notación musical.

`musicxml2ly` extrae las notas, articulaciones, estructura de la partitura y letra de archivos de MusicXML 'que estén divididos en partes', y los escribe en un archivo `.ly`. Se ejecuta desde de la línea de órdenes como sigue:

```
musicxml2ly [opción]... archivo.xml
```

Observe que por 'línea de órdenes', nos referimos a la línea de órdenes del sistema operativo. Consulte Sección 4.3 [Conversión desde otros formatos], página 47, para obtener más información acerca de esto.

Si el nombre de archivo proporcionado es `-` en lugar de *archivo.xml*, `musicxml2ly` lee toda la entrada directamente a partir de la entrada estándar.

`musicxml2ly` contempla las siguientes opciones:

`-a, --absolute`
: convertir las alturas en modo absoluto.

`--fb --fretboards`
: convierte los eventos `<frame>` en una voz independiente de diagramas de trastes en lugar de elementos de marcado.

`-h, --help`
: mostrar un resumen de la utilización y las opciones.

`-l, --language=IDIOMA`
: utilizar IDIOMA para los nombres de las notas, p.ej. `espanol` para los nombres de las notas en español.

`--loglevel=`*LOGLEVEL*
: fijar el grado en que la salida es prolija al valor dado en *LOGLEVEL*. Los valores posibles son `NONE` (ninguno), `ERROR` (errores), `WARNING` (advertencias), `PROGRESS` (avance; predeterminado) y `DEBUG` (depuración).

`--lxml` usar el paquete de Python lxml.etree para el análisis de XML; usa menos memoria y tiempo de CPU.

`-m, --midi`
activar el bloque midi dentro del archivo *.ly*.

`--nb, --no-beaming`
no convertir la información de las barras, en vez de ello usar el barrado automático de LilyPond.

`--nd, --no-articulation-directions`
no convertir las direcciones (^, _ o -) para las articulaciones, dinámica, etc.

`--nrp, --no-rest-positions`
no convertir las posiciones verticales exactas de los silencios.

`--nsb, --no-system-breaks`
ignorar los saltos de sistema.

`--npl, --no-page-layout`
no convertir la disposición exacta de la página y los saltos (es una forma compacta equivalente a las opciones `--nsb --npb --npm`).

`--npb, --no-page-breaks`
ignorar los saltos de página.

`--npm, --no-page-margins`
ignorar los márgenes de la página.

`--nsd, --no-stem-directions`
ignorar las direcciones de las plicas a partir del MusicXML, y en su lugar utilizar las plicas de dirección automática de LilyPond.

`-o, --output=`*ARCHIVO*
fijar el nombre del archivo de salida como *ARCHIVO*. Si *ARCHIVO* es -, la salida se imprime sobre stdout, la salida estándar. Si no se proporciona ninguno, en su lugar se usa *archivo_xml.ly*.

`-r, --relative`
convertir las alturas en modo relativo (predeterminado).

`--transpose=NOTA_DESTINO`
transporte que efectuar, entendido como el intervalo entre la nota `c` y *NOTA_DESTINO*.

`--sm, --shift-meter=BEATS/BEATTYPE`
cambiar la longitud o duración de las notas como una función de una indicación de compás dada para hacer que la partitura parezca más rápida o más lenta, (p.ej. `4/4` o `2/2`).

`--tc, --tab-clef=TABCLEFNAME`
alternar entre las dos versiones de claves de tablatura (`tab` y `moderntab`).

`--sn --string-numbers=t[rue]/f[alse]`
desactivar el símbolo del número de cuerda con `--string-numbers false`. Lo predeterminado es `true`.

`-v, --verbose`
ser prolijo.

`-v, --version`
mostrar la información de la versión y salir.

`-z, --compressed`
: el archivo de entrada es un archivo MusicXML comprimido en zip.

4.3.3 Invocar `abc2ly`

> **Nota:** Este programa ya no está soportado, y podría desaparecer de versiones posteriores de LilyPond.

ABC es un formato bastante simple basado en ASCII. Se encuentra descrito en el sitio web de ABC:

`http://www.walshaw.plus.com/abc/learn.html`.

`abc2ly` convierte ABC en LilyPond. Se invoca de la siguiente manera:

`abc2ly [`*opción*`]... `*archivo_abc*

`abc2ly` contempla las siguientes opciones:

`-b, --beams=None`
: preservar la noción de ABC de las barras

`-h, --help`
: esta ayuda

`-o, --output=`*archivo*
: fijar el nombre del archivo de salida como *archivo*.

`-s, --strict`
: ser estricto respecto al éxito

`--version`
: imprimir la información de la versión.

Existe una posibilidad rudimentaria para añadir código de LilyPond el archivo fuente de ABC. Por ejemplo:

```
%%LY voices \set autoBeaming = ##f
```

hará que el texto que sigue a la palabra clave 'voices' se inserte en la voz en curso del archivo de salida de LilyPond.

De forma similar,

```
%%LY slyrics más palabras
```

producirá que el texto que sigue a la palabra clave 'slyrics' se inserte en la línea de letra en curso.

Advertencias y problemas conocidos

El estándar ABC no es muy 'estándar'. Existen diferentes convenciones para las posibilidades avanzadas (por ejemplo, polifonía).

No se pueden convertir varias melodías de un solo archivo.

ABC sincroniza las letras y las notas al principio de una línea; `abc2ly` no lo hace.

`abc2ly` ignora el barrado de ABC.

4.3.4 Invocar `etf2ly`

> **Nota:** Este programa ya no está soportado y podría desaparecer de versiones posteriores de LilyPond.

ETF (Enigma Transport Format) es un formato utilizado por Finale, un producto de Coda Music Technology. `etf2ly` convierte parte de un archivo ETF en un archivo de LilyPond listo para usar.

Se invoca a través de la línea de órdenes como sigue:

```
etf2ly [opción]... archivo_etf
```

Observe que por 'línea de órdenes', nos referimos a la línea de órdenes del sistema operativo. Consulte Sección 4.3 [Conversión desde otros formatos], página 47, para obtener más información acerca de esto.

`etf2ly` contempla las siguientes opciones:

`-h, --help`
: esta ayuda

`-o, --output=`*ARCHIVO*
: fijar el nombre del archivo de salida como *ARCHIVO*

`--version`
: información de la versión

Advertencias y problemas conocidos

La lista de inscripciones de articulación posibles es incompleta. Los compases vacíos confunden a `etf2ly`. Las secuencias de notas de adorno no se dan por finalizadas satisfactoriamente.

4.3.5 Otros formatos

El propio LilyPond no contempla la utilización de ningún otro formato, pero existen algunas herramientas externas que también generan archivos de LilyPond. Se encuentran relacionados en la sección Sección "Entornos mejorados" en *Información general*.

4.4 Salida de LilyPond dentro de otros programas

Esta sección presenta métodos para integrar texto y música distintos del método automatizado con `lilypond-book`.

4.4.1 LuaTEX

Además de `lilypond-book` para integrar la salida de LilyPond, existe un programa alternativo que puede utilizarse si se emplea LuaTEX, llamado lyluatex (`https://github.com/jperon/lyluatex/blob/master/README.en.md`).

4.4.2 OpenOffice y LibreOffice

Se puede añadir notación de LilyPond a los documentos de OpenOffice.org y LibreOffice con OOoLilyPond (`https://github.com/openlilylib/LO-ly`), una extensión de OpenOffice.org que convierte archivos de LilyPond en imágenes dentro de los documentos de OpenOffice.org. OOoLilyPond (OLy) funciona con versiones recientes de LibreOffice y OpenOffice. También deben funcionar las versiones antiguas. Incluso se ha comprobado que funciona con OpenOffice 2.4 sin ningún problema.

4.4.3 Otros programas

Otros programas capaces de manejra los formatos `PNG`, `EPS` o `PDF` deberían usar `lilypond` en vez de `lilypond-book`. Cada archivo de salida de LilyPond debe crearse individualmente y añadirse al documento; consulte la documentación del programa correspondiente acerca de la manera de insertar archivos desde otras fuentes.

Para reducir el espacio vacío alrededor de la partitura de LilyPond, utilice las siguientes opciones:

```
\paper{
  indent=0\mm
  line-width=120\mm
  oddFooterMarkup=##f
  oddHeaderMarkup=##f
  bookTitleMarkup = ##f
  scoreTitleMarkup = ##f
}

... music ...
```

Para producir imágenes EPS:

```
lilypond -dbackend=eps -dno-gs-load-fonts -dinclude-eps-fonts miarchivo.ly
```

Para producir imágenes PNG:

```
lilypond -dbackend=eps -dno-gs-load-fonts -dinclude-eps-fonts --png miarchivo.ly
```

Para producir imágenes PNG con transparencia:

```
lilypond -dbackend=eps -dno-gs-load-fonts -dinclude-eps-fonts -dpixmap-format=pngalpha
```

Si necesita citar muchos fragmentos de una partitura grande, también puede usar la funcionalidad clip-systems de recorte de sistemas, véase Sección “Extracción de fragmentos de música” en *Referencia de la Notación*.

4.5 Archivos de inclusión independientes

Algunos usuarios han producido archivos que se pueden incluir con la instrucción `\include` en LilyPond para producir ciertos efectos, y aquellos que se relacionan más abajo forman parte de la distribución de LilyPond. Véase también Sección “Trabajar sobre los archivos de entrada” en *Referencia de la Notación*.

4.5.1 Articulación MIDI

El proyecto Articulate (`http://www.nicta.com.au/articulate`) es un intento de mejora de la salida MIDI de LilyPond, y funciona ajustando la duración de las notas (que no estén bajo ligaduras de expresión) de acuerdo con las articulaciones que lleve cada una. Por ejemplo, un ‘staccato’ reduce la duración a la mitad, ‘tenuto’ da a una nota la duración completa, etcétera. Véase Sección “Enriquecimiento de la salida MIDI” en *Referencia de la Notación*.

5 Sugerencias para escribir archivos de entrada

En este momento está preparado para comenzar a escribir archivos de LilyPond más grandes – no sólo los pequeños ejemplos que aparecen en el tutorial, sino piezas completas –. Pero ¿cómo debe proceder para hacerlo?

En la medida en que LilyPond entienda sus archivos y produzca la salida que usted pretendía, realmente no importa mucho qué aspecto tengan sus archivos. Sin embargo existen algunas otras cosas a tener en cuenta cuando se escriben archivos de LilyPond.

- ¿Qué ocurre si comete un fallo? La estructura de un archivo de LilyPond puede hacer que ciertos errores se hagan más fáciles (o más difíciles) de encontrar.
- ¿Qué ocurre si quiere compartir sus archivos con otras personas? De hecho, ¿y si quiere alterar sus propios archivos después de algunos años? Algunos archivos de LilyPond se comprenden a primera vista; otros pueden tenerle rascándose la cabeza durante una hora.
- ¿Qué ocurre si quiere actualizar su archivo de LilyPond para poderlo usar con una versión más reciente del programa?

  La sintaxis de la entrada se modifica de forma ocasional según LilyPond se va perfeccionando. Casi todos los cambios se pueden hacer de forma automática con `convert-ly`, pero algunos podrían necesitar de una ayuda manual. Los archivos de LilyPond se pueden estructurar para que sean más fáciles (o más difíciles) de actualizar.

5.1 Sugerencias de tipo general

Presentamos algunas sugerencias que le pueden servir de ayuda para evitar o corregir los problemas más comunes al realizar trabajos de tipografía musical:

- **Incluya siempre el número de `\version` en los archivos de entrada**, aun en los más pequeños. Ello evita tener que recordar para qué versión de LilyPond se creó el archivo y es especialmente relevante al Capítulo 2 [Actualizar ficheros con `convert-ly`], página 19, (una instrucción que requiere que el enunciado `\version` esté presente); o si está enviando código de entrada a otros usuarios (p.ej. si está pidiendo ayuda en una de las listas de distribución de correo). Observe que todas las plantillas de LilyPond contienen números de `\version`.
- **Escriba un compás de música en cada línea del código de entrada**. Esto hará que la búsqueda de problemas dentro de los archivos de entrada sea mucho más sencilla.
- **Inserte barras de Sección "Comprobación de compás y de número de compás" en** *Referencia de la Notación* **así como códigos de Sección "Comprobación de octava" en** *Referencia de la Notación*. La inclusión de códigos de comprobación de estos tipos será de ayuda para localizar los errores mucho más rápidamente. La frecuencia con que añadir las comprobaciones dependerá de la complejidad de la música que se está componiendo tipográficamente. Para composiciones sencillas, las comprobaciones añadidas en ciertos puntos estratégicos dentro de la música pueden ser suficientes, pero para música más compleja, con muchas voces y/o pentagramas, sería mejor poner comprobaciones a cada compás.
- **Inserte comentarios en el código de entrada**. Las referencias a los temas musicales (p.ej. 'segundo tema en los violines', 'cuarta variación', etc.), o simplemente la inclusión de los números de compás como compentarios, hará mucho más sencilla la navegación por el archivo de entrada, especialmente si más tarde se hace necesario alterar algo, o si estamos pasando los archivos de entrada a otra persona.
- **Escriba las duraciones explícitamente** al comienzo de las 'secciones'. Por ejemplo, si especifica `c4 d e f` al principio de una frase (en lugar de sólo `c d e f`) se puede ahorrar problemas si reelabora la música más tarde.
- **Aprenda a aplicar márgenes y sangrados a las llaves y a la música paralela**. Muchos problemas suelen estar producidos por llaves de apertura o de cierre que faltan. La aplicación

clara de sangrados a las llaves curvas de apertura y de cierre (o a los indicadores << y >>) será de ayuda para evitar tales problemas.

Por ejemplo:

```
\new Staff {
  \relative {
    r4 g'8 g c8 c4 d |
    e4 r8 |
    % sección de Ossia
    <<
      { f8 c c | }
      \new Staff {
        f8 f c |
      }
    >>
    r4 |
  }
}
```

es mucho más fácil de seguir que:

```
\new Staff { \relative { r4 g'8 g c4 c8 d | e4 r8
% sección de Ossia
<< { f8 c c } \new Staff { f8 f c } >> r4 | } }
```

- **Mantenga separados la música y el estilo** poniendo las sobreescrituras dentro del bloque `\layout`:

```
\score {
  ...música...
  \layout {
   \override TabStaff.Stemstencil = ##f
 }
}
```

Esto no crea un contexto nuevo, sino que se aplicará en el momento de crear uno. Véase también Sección "Ahorrar tecleo mediante variables y funciones" en *Manual de Aprendizaje* y Sección "Hojas de estilo" en *Manual de Aprendizaje*.

5.2 Tipografiar música existente

Si está introduciendo música a partir de una partitura existente (es decir, tipografiando una hoja de música ya impresa),

- Introduzca en LilyPond un sistema del manuscrito, o copia física, de cada vez (pero mantenga la práctica de escribir un compás por línea de texto), y compruebe cada sistema cuando lo haya terminado. Puede usar las propiedades `showLastLength` o `showFirstLength` para acelerar el proceso (véase Sección "Saltar la música corregida" en *Referencia de la Notación*).
- Defina `mBreak = { \break }` e inserte `\mBreak` dentro del archivo de entrada donde el manuscrito tenga un saldo de línea. De esta forma le resultará mucho más fácil comparar la música de LilyPond con la original. Cuando haya terminado de revisar su partitura podrá definir `mBreak = { }` para quitar todos esos saltos de línea. Así permitirá a LilyPond colocar los saltos donde éste lo estime más oportuno.
- Al escribir una parte para un instrumento transpositor dentro de una variable, se recomienda que las notas estén envueltas dentro de

```
\transpose c altura-natural {...}
```

(donde `altura-natural` es la afinación natural del instrumento) de forma que la música dentro de la variable esté realmente en Do mayor. Después podemos volver a transportarlas en sentido inverso cuando se utiliza la variable, si es necesario, pero quizá no queramos hacerlo (p.ej., al imprimir una partitura en afinación de concierto, al convertir una parte de trombón de clave de Sol a clave de Fa, etc.). Es menos probable cometer errores en los transportes si toda la música que está dentro de las variables se encuentra en un tono coherente.

Asimismo, haga los transportes exclusivamente hacia o desde Do mayor. Esto significa que aparte de ésta, las únicas tonalidades que usaremos serán los tonos de afinación de los instrumentos transpositores: bes para una trompeta en Si bemol, aes para un clarinete en La bemol, etc.

5.3 Proyectos grandes

Al trabajar en proyectos grandes se hace esencial tener una estructura clara en los archivos de LilyPond:

- **Utilice un identificador para cada voz**, con un mínimo de estructura dentro de la definición. La estructura de la sección `\score` es la que cambiará con mayor probabilidad; por contra, es extremadamente improbable que cambie la definición de `violin` en versiones nuevas de LilyPond.

```
violin = \relative {
g'4 c'8. e16
}
...
\score {
  \new GrandStaff {
    \new Staff {
      \violin
    }
  }
}
```

- **Separe los trucos de las definiciones musicales**. Ya se mencionó con anterioridad, pero para proyectos grandes es vital. Quizá tengamos que cambiar la definición de `fluegop`, pero en ese caso sólo lo tendremos que hacer una vez, y aún podremos evitar tocar nada dentro de `violin`.

```
fluegop = _\markup{
  \dynamic f \italic \small { 2nd } \hspace #0.1 \dynamic p }
violin = \relative {
g'4\fluegop c'8. e16
}
```

5.4 Solución de problemas

Antes o después escribirá un archivo que LilyPond no podrá compilar. Los mensajes que LilyPond proporciona pueden ayudarle a encontrar el error, pero en muchos casos tendrá que llevar a cabo algún tipo de investigación para determinar el origen del problema.

Las herramientas más poderosas para este cometido son el comentario de una sola línea (indicado por %) y el comentario de bloque (indicado por %{ . . .%}). Si no sabe dónde está el problema, comience convirtiendo grandes secciones del archivo de entrada en un comentario. Después de eliminar una sección convirtiéndola en un comentario, pruebe a compilar el archivo otra vez. Si funciona, entonces el problema debía estar en la porción que había eliminado. Si no

funciona, continúe eliminando material (transformándolo en comentarios) hasta que tenga algo que funcione.

En un caso extremo podría terminar con sólo

```
\score {
  <<
    % \melodia
    % \armonia
    % \bajo
  >>
  \layout{}
}
```

(en otras palabras: un archivo sin música)

Si ocurre esto, no abandone. Descomente un trozo pequeño – digamos la parte del bajo – y observe si funciona. Si no es así, transforme en comentarios toda la música del bajo (pero deje el `\bajo` de la sección `\score` no comentado.

```
bajo = \relative {
%{
  c'4 c c c
  d d d d
%}
}
```

Ahora empiece poco a poco descomentando cada vez más fracciones de la parte del `bajo` hasta que encuentre la línea del problema.

Otra técnica de depuración muy útil es la construcción de Sección “Ejemplos mínimos” en *Información general*.

5.5 Make y los Makefiles

Posiblemente todas las plataformas en que puede correr LilyPond, contemplan una posibilidad de software llamada `make`. Este programa lee un archivo especial llamado `Makefile` que define las relaciones de dependencia entre los archivos y qué instrucciones necesitamos dar al sistema operativo para producir un archivo a partir de otro. Por ejemplo, el archivo de make detallaría cómo obtener `balada.pdf` y `balada.midi` a partir de `balada.ly` mediante la ejecución de LilyPond.

Existen ocasiones en las que es buena idea crear un `Makefile` para nuestro proyecto, bien sea por nuestra propia comodidad o como cortesía para otros que posiblemente tengan acceso a nuestros archivos fuente. Esto es cierto para proyectos muy grandes con muchos archivos de inclusión y distintas opciones de salida (p.ej. partitura completa, particellas, partitura del director, reducción para piano, etc.), o para proyectos que requieren instrucciones difíciles para montarlas (como los proyectos de `lilypond-book`). La complejidad y flexibilidad de los Makefiles varía enormemente según las necesidades y la habilidad de los autores. El programa GNU Make viene instalado en las distribuciones de GNU/Linux y en MacOS X, y también existe para Windows.

Consulte el **Manual de GNU Make** para ver todos los detalles sobre el uso de `make`, pues lo que sigue a continuación ofrece solamente una pincelada de todo lo que es capaz de hacer.

Las instrucciones que definen las reglas en un archivo de make difieren en función de la plataforma; por ejemplo, las distintas formas de GNU/Linux y MacOS usan `bash`, mientras que Windows usa `cmd`. Observeque en MacOS X, tenemos que configurar el sistema para que utilice el intérprete de órdenes. A continuación presentamos algunos makefiles de ejemplo, con versiones tanto para GNU/Linux/MacOS como para Windows.

El primer ejemplo es para una obra orquestal en cuatro movimientos con la estructura de directorios siguiente:

```
Sinfonia/
|-- MIDI/
|-- Makefile
|-- Notas/
|   |-- cello.ily
|   |-- cifras.ily
|   |-- trompa.ily
|   |-- oboe.ily
|   |-- trioCuerdas.ily
|   |-- viola.ily
|   |-- violinUno.ily
|   `-- violinDos.ily
|-- PDF/
|-- Particellas/
|   |-- sinfonia-cello.ly
|   |-- sinfonia-trompa.ly
|   |-- sinfonia-oboes.ly
|   |-- sinfonia-viola.ly
|   |-- sinfonia-violinUno.ly
|   `-- sinfonia-violinDos.ly
|-- Partituras/
|   |-- sinfonia.ly
|   |-- sinfoniaI.ly
|   |-- sinfoniaII.ly
|   |-- sinfoniaIII.ly
|   `-- sinfoniaIV.ly
`-- sinfoniaDefs.ily
```

Los archivos `.ly` de los directorios `Partituras` y `Particellas` obtienen las notas de archivos `.ily` que están en el directorio `Notas`:

```
%%% principio del archivo "sinfonia-cello.ly"
\include ../definicionesSinf.ily
\include ../Notas/cello.ily
```

El makefile tendrá los objetivos de `partitura` (la pieza completa en todo su esplendor), `movimientos` (partitura completa de los movimientos individuales) y `particellas` (partes individuales para los atriles). También existe un objetivo `archivo` que produce un tarball de los archivos fuente, adecuado para compartirlo a través de la web o por correo electrónico. A continuación presentamos el makefile para GNU/Linux o MacOS X. Se debe guardar con el nombre exacto `Makefile` el el directorio superior del proyecto:

> **Nota:** Cuando se define un objetivo o una regla de patrón, las líneas siguientes deben comenzar con tabuladores, no con espacios.

```
# nombre principal de los archivos de salida
nombre = sinfonia
# determinar cuántos procesadores existen
CPU_CORES=`cat /proc/cpuinfo | grep -m1 "cpu cores" | sed s/".*: "//`
# La instrucción para ejecutar lilypond
LILY_CMD = lilypond -ddelete-intermediate-files \
                    -dno-point-and-click -djob-count=$(CPU_CORES)
```

```
# Los sufijos utilizados en este Makefile.
.SUFFIXES: .ly .ily .pdf .midi

# Los archivos de entrada y salida se buscan dentro de los directorios relacionados en
# la variable VPATH.  Todos esllos son subdirectorios del directorio
# en curso (dado por la variable de GNU make `CURDIR').
VPATH = \
  $(CURDIR)/Partituras \
  $(CURDIR)/PDF \
  $(CURDIR)/Particellas \
  $(CURDIR)/Notas

# La regla de patrón para crear archivos PDF y MIDI a partir de un archivo de entrada l
# Los archivos de salida .pdf se colocan en el subdirectorio `PDF', y los archivos
# .midi van al subdirectorio `MIDI'.
%.pdf %.midi: %.ly
        $(LILY_CMD) $<; \           # esta línea comienza con un salto de tabulación
        if test -f "$*.pdf"; then \
            mv "$*.pdf" PDF/; \
        fi; \
        if test -f "$*.midi"; then \
            mv "$*.midi" MIDI/; \
        fi

notas = \
  cello.ily \
  trompa.ily \
  oboe.ily \
  viola.ily \
  violinUno.ily \
  violinDos.ily

# Dependencias de los movimientos.
$(nombre)I.pdf: $(nombre)I.ly $(notas)
$(nombre)II.pdf: $(nombre)II.ly $(notas)
$(nombre)III.pdf: $(nombre)III.ly $(notas)
$(nombre)IV.pdf: $(nombre)IV.ly $(notas)

# Dependencias de la partitura completa.
$(nombre).pdf: $(nombre).ly $(notas)

# Dependencias de las particellas.
$(nombre)-cello.pdf: $(nombre)-cello.ly cello.ily
$(nombre)-trompa.pdf: $(nombre)-trompa.ly trompa.ily
$(nombre)-oboes.pdf: $(nombre)-oboes.ly oboe.ily
$(nombre)-viola.pdf: $(nombre)-viola.ly viola.ily
$(nombre)-violinUno.pdf: $(nombre)-violinUno.ly violinUno.ily
$(nombre)-violinDos.pdf: $(nombre)-violinDos.ly violinDos.ily

# Teclee `make partitura' para generer la partitura completa de los cuatro
# movimientos como un archivo único.
```

```
.PHONY: partitura
partitura: $(nombre).pdf

# Teclee `make particellas' para generar todas las particellas.
# Teclee `make fulanito.pdf' para generar la particella del instrumento `fulanito'.
# Ejemplo: `make sinfonia-cello.pdf'.
.PHONY: particellas
particellas: $(nombre)-cello.pdf \
       $(nombre)-violinUno.pdf \
       $(nombre)-violinDos.pdf \
       $(nombre)-viola.pdf \
       $(nombre)-oboes.pdf \
       $(nombre)-trompa.pdf

# Teclee `make movimientos' para generar los archivos de los
# cuatro movimientos de forma separada.
.PHONY: movimientos
movimientos: $(nombre)I.pdf \
           $(nombre)II.pdf \
           $(nombre)III.pdf \
           $(nombre)IV.pdf

all: partitura particellas movimientos

archivo:
        tar -cvvf stamitz.tar \       # esta línea comienza con un salto de tabulación
        --exclude=*pdf --exclude=*~ \
        --exclude=*midi --exclude=*.tar \
        ../Stamitz/*
```

Existen ciertas complicaciones en la plataforma Windows. Después de descargar e instalar el programa GNU Make para Windows, debemos configurar la ruta adecuada en las variables de entorno del sistema de forma que el shell del DOS pueda encontrar el programa Make. Para hacerlo, pulse con el botón derecho sobre "Mi PC", elija `Propiedades` y `Avanzadas`. Pulse sobre `Variables de entorno`, y luego en la pestaña `Variables del sistema`, seleccione `Ruta`, pulse sobre `editar` y añada la ruta al archivo ejecutable de GNU Make, con lo que quedará algo parecido a lo siguiente:

```
C:\Archivos de programa\GnuWin32\bin
```

El makefile en sí debe modificarse para que maneje distintas instrucciones del shell y para que pueda tratar con los espacios que aparecen en el nombre de algunos directorios del sistema predeterminados. El objetivo `archivo` se elimina porque Windows no tiene la instrucción `tar`, y Windows tiene también una extensión predeterminada distinta para los archivos MIDI.

```
## VERSIÓN PARA WINDOWS
##
nombre = sinfonia
LILY_CMD = lilypond -ddelete-intermediate-files \
                    -dno-point-and-click \
                    -djob-count=$(NUMBER_OF_PROCESSORS)

#obtener el nombre 8.3 de CURDIR (rodeo para los espacios en PATH)
workdir = $(shell for /f "tokens=*" %%b in ("$(CURDIR)") \
          do @echo %%~sb)
```

```
.SUFFIXES: .ly .ily .pdf .mid

VPATH = \
  $(workdir)/Partituras \
  $(workdir)/PDF \
  $(workdir)/Particellas \
  $(workdir)/Notas

%.pdf %.mid: %.ly
        $(LILY_CMD) $<      # esta línea comienza con un salto de tabulación
        if exist "$*.pdf"  move /Y "$*.pdf"  PDF/ # comienzo con tab
        if exist "$*.mid" move /Y "$*.mid" MIDI/  # comienzo con tab

notas = \
  cello.ily \
  cifras.ily \
  trompa.ily \
  oboe.ily \
  trioCuerdas.ily \
  viola.ily \
  violinUno.ily \
  violinDos.ily

$(nombre)I.pdf: $(nombre)I.ly $(notas)
$(nombre)II.pdf: $(nombre)II.ly $(notas)
$(nombre)III.pdf: $(nombre)III.ly $(notas)
$(nombre)IV.pdf: $(nombre)IV.ly $(notas)

$(nombre).pdf: $(nombre).ly $(notas)

$(nombre)-cello.pdf: $(nombre)-cello.ly cello.ily
$(nombre)-trompa.pdf: $(nombre)-trompa.ly trompa.ily
$(nombre)-oboes.pdf: $(nombre)-oboes.ly oboe.ily
$(nombre)-viola.pdf: $(nombre)-viola.ly viola.ily
$(nombre)-violinUno.pdf: $(nombre)-violinUno.ly violinUno.ily
$(nombre)-violinDos.pdf: $(nombre)-violinDos.ly violinDos.ily

.PHONY: partitura
partitura: $(nombre).pdf

.PHONY: particellas
particellas: $(nombre)-cello.pdf \
       $(nombre)-violinUno.pdf \
       $(nombre)-violinDos.pdf \
       $(nombre)-viola.pdf \
       $(nombre)-oboes.pdf \
       $(nombre)-trompa.pdf

.PHONY: movimientos
movimientos: $(nombre)I.pdf \
           $(nombre)II.pdf \
```

```
            $(nombre)III.pdf \
            $(nombre)IV.pdf

all: partitura particellas movimientos
```

El Makefile siguiente es para un documento de `lilypond-book` hecho en LaTeX. Este proyecto tiene un índice, que requiere ejecutar la instrucción `latex` dos veces para actualizar los enlaces. Todos los archivos de salida se almacenan en el directorio `salida` para los documentos .pdf y en el directorio `salidahtml` para la salida en formato html.

```
SHELL=/bin/sh
NOMBRE=miproyecto
DIR_SALIDA=salida
DIR_WEB=salidahtml
VISOR=acroread
NAVEGADOR=firefox
LILYBOOK_PDF=lilypond-book --output=$(DIR_SALIDA) --pdf $(NOMBRE).lytex
LILYBOOK_HTML=lilypond-book --output=$(DIR_WEB) $(NOMBRE).lytex
PDF=cd $(DIR_SALIDA) && pdflatex $(NOMBRE)
HTML=cd $(DIR_WEB) && latex2html $(NOMBRE)
INDICE=cd $(DIR_SALIDA) && makeindex $(NOMBRE)
VISTA_PREVIA=$(VISOR) $(DIR_SALIDA)/$(NOMBRE).pdf &

all: pdf web guardar

pdf:
        $(LILYBOOK_PDF)  # comienza con un tab
        $(PDF)           # comienza con un tab
        $(INDICE)        # comienza con un tab
        $(PDF)           # comienza con un tab
        $(VISTA_PREVIA)  # comienza con un tab

web:
        $(LILYBOOK_HTML) # comienza con un tab
        $(HTML)          # comienza con un tab
        cp -R $(DIR_WEB)/$(NOMBRE)/ ./  # comienza con un tab
        $(NAVEGADOR) $(NOMBRE)/$(NOMBRE).html &  # comienza con un tab

guardar: pdf
        cp $(DIR_SALIDA)/$(NOMBRE).pdf $(NOMBRE).pdf  # comienza con un tab

clean:
        rm -rf $(DIR_SALIDA) # comienza con un tab

web-clean:
        rm -rf $(DIR_WEB) # comienza con un tab

archivo:
        tar -cvvf miproyecto.tar \ # comienza con un tab
        --exclude=salida/* \
        --exclude=salidahtml/* \
        --exclude=miproyecto/* \
        --exclude=*midi \
```

```
--exclude=*pdf \
--exclude=*~ \
../MiProyecto/*
```

HACER: conseguir que funcione en Windows

El makefile anterior no funciona en Windows. Una alternativa para los usuarios de Windows sería crear un archivo de lotes sencillo que contenga las instrucciones de montaje. Esto no rastrea las dependencias en la manera en que lo hace un makefile, pero al menos reduce el proceso de construcción a una sola instrucción. Guarde el código siguiente como `montaje.bat` o `montaje.cmd`. El archivo de lotes se puede ejecutar en la línea de órdenes del DOS o simplemente haciendo doble click sobre su icono.

```
lilypond-book --output=salida --pdf miproyecto.lytex
cd salida
pdflatex miproyecto
makeindex miproyecto
pdflatex miproyecto
cd ..
copy salida\miproyecto.pdf MiProyecto.pdf
```

Véase también

Manual de utilización del programa: Sección “Utilización desde la línea de órdenes” en *Utilización del Programa*, Sección “lilypond-book” en *Utilización del Programa*

Apéndice A GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both

covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

   A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
   B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
   C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
   D. Preserve all the copyright notices of the Document.
   E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
   F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
   G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
   H. Include an unaltered copy of this License.
   I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its

Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `https://www.gnu.org/licenses/`.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Apéndice B Índice de LilyPond

P

Q

R

S

T

U

V

W

X